

# Package ‘debrowser’

April 11, 2018

**Type** Package

**Title** debrowser: Interactive Differential Expression Analysis Browser

**Version** 1.6.8

**Date** 2016-07-05

**Author** Alper Kucukural <alper.kucukural@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, users can easily discover different parts of the data that like never have been done before. Manually creating and looking these plots takes time. With DEBrowser 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, gplots, 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, shinydashboard, RCurl, pathview, shinyBS, googleAuthR

**RoxygenNote** 6.0.1

**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:**

actionButton . . . . .	4
addDataCols . . . . .	5
addID . . . . .	6
add_title_pos . . . . .	6
all2all . . . . .	7
applyFilters . . . . .	7
applyFiltersToMergedComparison . . . . .	8
bookmarkServer . . . . .	9
bookmarkUI . . . . .	9
clusterData . . . . .	10
compareClust . . . . .	10
copy2newDirectory . . . . .	11
correctBatchEffect . . . . .	11
deServer . . . . .	12
deUI . . . . .	13
drawPCAExplained . . . . .	13
getAfterLoadMsg . . . . .	14
getColors . . . . .	14
getColorShapeSelection . . . . .	15
getCompSelection . . . . .	15
getConditionSelector . . . . .	16
getConditionSelectorFromMeta . . . . .	16
getCondMsg . . . . .	17
getCutOffSelection . . . . .	17
getDataForTables . . . . .	18
getDataPrepPanel . . . . .	19
getDensityPlot . . . . .	19
getDomains . . . . .	20
getDown . . . . .	20
getDownloadSection . . . . .	21
getEnrichDO . . . . .	21
getEnrichGO . . . . .	22
getEnrichKEGG . . . . .	23
getEntrezIds . . . . .	23
getGeneList . . . . .	24
getGeneSetData . . . . .	25
getGOLeftMenu . . . . .	25
getGoPanel . . . . .	26
getGOPlots . . . . .	26
getHelpButton . . . . .	27
getHoverPlots . . . . .	27
getInitialMenu . . . . .	28
getIntHeatmap . . . . .	28
getIntHeatmapVis . . . . .	29
getIQRPlot . . . . .	29
getJsonObj . . . . .	30
getKEGGModal . . . . .	30
getLeftMenu . . . . .	31
getLegendSelect . . . . .	31
getLoadingMsg . . . . .	32

getLogo . . . . .	32
getMainPanel . . . . .	33
getMainPanelPlots . . . . .	33
getMainPlotsLeftMenu . . . . .	34
getMean . . . . .	34
getMergedComparison . . . . .	35
getMethodDetails . . . . .	35
getMostVariedList . . . . .	36
getNormalizedMatrix . . . . .	36
getOrganism . . . . .	37
getOrganismBox . . . . .	37
getOrganismPathway . . . . .	38
getPCAexplained . . . . .	38
getPCselection . . . . .	39
getProgramTitle . . . . .	40
getQCLeftMenu . . . . .	40
getQCPanel . . . . .	41
getQCPlots . . . . .	41
getQCReplot . . . . .	42
getSampleNames . . . . .	43
getSamples . . . . .	43
getSearchData . . . . .	44
getSelectedCols . . . . .	44
getSelectedDatasetInput . . . . .	45
getSelectInputBox . . . . .	45
getSelHeat . . . . .	46
getShapeColor . . . . .	47
getStartPlotsMsg . . . . .	47
getStartupMsg . . . . .	48
getTableStyle . . . . .	48
getTextOnOff . . . . .	49
getToolTipPCA . . . . .	49
getToolTipText . . . . .	50
getUp . . . . .	50
getUpDown . . . . .	51
get_state_id . . . . .	51
get_user_info . . . . .	52
hideObj . . . . .	52
installpack . . . . .	53
link_brush . . . . .	53
loadpack . . . . .	54
load_data . . . . .	54
logSliderJScode . . . . .	55
mainScatter . . . . .	55
MAPlot . . . . .	56
MAZoom . . . . .	57
panel.cor . . . . .	57
panel.hist . . . . .	58
plot_pca . . . . .	58
prepAddQCPlots . . . . .	59
prepDataContainer . . . . .	60
prepDataForQC . . . . .	60

prepDEOutput . . . . .	61
push . . . . .	61
readMetaData . . . . .	62
removeBookmark . . . . .	62
removeCols . . . . .	63
round_vals . . . . .	63
runBayseq . . . . .	64
runDE . . . . .	64
runDESeq . . . . .	65
runDESeq2 . . . . .	66
runEdgeR . . . . .	67
runHeatmap . . . . .	68
runLimma . . . . .	68
run_pca . . . . .	69
saveQCPlot . . . . .	70
scatterZoom . . . . .	71
selectBatchEffect . . . . .	71
selectConditions . . . . .	72
selectedInput . . . . .	73
setFilterParams . . . . .	73
showObj . . . . .	74
startDEBrowser . . . . .	74
textareaInput . . . . .	75
togglePanels . . . . .	75
volcanoPlot . . . . .	76
volcanoZoom . . . . .	76
whitelist . . . . .	77

<b>Index</b>	<b>78</b>
--------------	-----------

---

actionButton	<i>Buttons including Action Buttons and Event Buttons</i>
--------------	---

---

## Description

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

## Usage

```
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

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

---

addDataCols	<i>addDataCols</i>
-------------	--------------------

---

### Description

add additional data columns to de results

### Usage

```
addDataCols(data = NULL, de_res = NULL, cols = NULL, inputconds = NULL,  
i = NULL, input = NULL)
```

### Arguments

data,	loaded dataset
de_res,	de results
cols,	columns
inputconds,	inputconds
i,	selected comparison number
input,	input

### Value

data

### Examples

```
x <- addDataCols()
```

---

addID	<i>addID</i>
-------	--------------

---

**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	<i>add_title_pos</i>
---------------	----------------------

---

**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**

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

---

all2all

*all2all*


---

**Description**

Prepares all2all scatter plots for given datasets.

**Usage**

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

**Arguments**

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

**Value**

all2all scatter plots

**Examples**

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

---

applyFilters

*applyFilters*


---

**Description**

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

**Usage**

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

**Arguments**

<code>filt_data,</code>	loaded dataset
<code>cols,</code>	selected samples
<code>conds,</code>	selected conditions
<code>input,</code>	input parameters

**Value**

data

**Examples**

```
x <- applyFilters()
```

---

`applyFiltersToMergedComparison`

*applyFiltersToMergedComparison*

---

**Description**

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

**Usage**

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

**Arguments**

<code>merged,</code>	merged data
<code>nc,</code>	the number of comparisons
<code>input,</code>	input params

**Value**

data

**Examples**

```
x <- applyFiltersToMergedComparison()
```



---

bookmarkServer	<i>bookmarkServer</i>
----------------	-----------------------

---

**Description**

bookmark Server functions

**Usage**

```
bookmarkServer(input = NULL, output = NULL, session = NULL,  
               loadingJSON = NUL)
```

**Arguments**

input,	input
output,	output
session,	session
loadingJSON,	loadingJSON

**Examples**

```
x <- bookmarkServer()
```

---

bookmarkUI	<i>bookmarkUI</i>
------------	-------------------

---

**Description**

bookmark UI

**Usage**

```
bookmarkUI(id = NULL)
```

**Arguments**

id,	id
-----	----

**Examples**

```
x <- bookmarkUI()
```

---

clusterData	<i>clusterData</i>
-------------	--------------------

---

**Description**

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

**Usage**

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

**Arguments**

dat,                   the data to cluster

**Value**

clustered data

**Note**

clusterData

**Examples**

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

---

compareClust	<i>compareClust</i>
--------------	---------------------

---

**Description**

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

**Usage**

```
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

**Value**

compared cluster

**Note**

compareClust

**Examples**

```
x <- compareClust()
```

---

copy2newDirectory	<i>copy2newDirectory</i>
-------------------	--------------------------

---

**Description**

To copy the bookmarked folder into a user named directory

**Usage**

```
copy2newDirectory(new_state_id = NULL, username = NULL, session = NULL)
```

**Arguments**

new_state_id,	new state id
username,	username
session,	session

**Examples**

```
x <- copy2newDirectory()
```

---

correctBatchEffect	<i>Correct Batch Effect</i>
--------------------	-----------------------------

---

**Description**

Batch effect correction

**Usage**

```
correctBatchEffect(idata = NULL, input = NULL)
```

**Arguments**

idata,	data
input,	input values

**Value**

data

**Examples**

```
x<-correctBatchEffect ()
```

---

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*

---

**Description**

Creates a shinyUI to be able to run DEBrowser interactively.

**Usage**

```
deUI()
```

**Value**

the panel for main plots;

**Note**

deUI

**Examples**

```
x<-deUI()
```

---

`drawPCAExplained`*drawPCAExplained*

---

**Description**

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

**Usage**

```
drawPCAExplained(explainedData = NULL)
```

**Arguments**

```
explainedData,  
selected data
```

**Value**

explained plot

**Examples**

```
x <- drawPCAExplained()
```

getAfterLoadMsg      *getAfterLoadMsg*

---

**Description**

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

**Usage**

```
getAfterLoadMsg()
```

**Value**

return After Load Msg

**Note**

```
getAfterLoadMsg
```

**Examples**

```
x <- getAfterLoadMsg()
```

---

getColors      *getColors*

---

**Description**

get colors for the domains

**Usage**

```
getColors(domains = NULL)
```

**Arguments**

domains,      domains to be colored

**Value**

colors

**Examples**

```
x<-getColors()
```

---

`getColorShapeSelection`      *getColorShapeSelection*

---

**Description**

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

**Usage**

```
getColorShapeSelection(input = NULL)
```

**Arguments**

input,                  input values

**Value**

Color and shape selection boxes

**Examples**

```
x <- getColorShapeSelection()
```

---

`getCompSelection`      *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`    *getConditionSelector*

---

**Description**

Selects user input conditions to run in DESeq.

**Usage**

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

**Arguments**

<code>num</code> ,	panel that is going to be shown
<code>choices</code> ,	sample list
<code>selected</code> ,	selected sample list

**Examples**

```
x <- getConditionSelector()
```

---

`getConditionSelectorFromMeta`  
*getConditionSelectorFromMeta*

---

**Description**

Selects user input conditions to run in DESeq from metadata

**Usage**

```
getConditionSelectorFromMeta(input = NULL, index = 1, num = 0,  
  choices = NULL, selected = NULL, loadingJSON = NULL)
```

**Arguments**

<code>input</code> ,	input
<code>index</code> ,	index
<code>num</code> ,	num
<code>choices</code> ,	choices
<code>selected</code> ,	selected
<code>loadingJSON</code> ,	loadingJSON

**Examples**

```
x <- getConditionSelectorFromMeta()
```



---

getCondMsg	<i>getCondMsg</i>
------------	-------------------

---

**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	<i>getCutOffSelection</i>
--------------------	---------------------------

---

**Description**

Gathers the cut off selection for DE analysis

**Usage**

```
getCutOffSelection(nc = 1)
```

**Arguments**

nc,	total number of comparisons
-----	-----------------------------

**Value**

returns the left menu according to the selected tab;

**Note**

getCutoffSelection

**Examples**

```
x <- getCutoffSelection()
```

---

getDataForTables      *getDataForTables get data to fill up tables tab*

---

**Description**

getDataForTables get data to fill up tables tab

**Usage**

```
getDataForTables(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()
```

`getDataPrepPanel`      *getDataPrepPanel*

**Description**

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

**Usage**

```
getDataPrepPanel(flag = FALSE)
```

**Arguments**

`flag`,                      flag to show the element in the ui

**Value**

returns the left menu according to the selected tab;

**Note**

`getDataPrepPanel`

**Examples**

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

`getDensityPlot`              *getDensityPlot*

**Description**

Makes Density plots

**Usage**

```
getDensityPlot(data = NULL, cols = NULL, title = "")
```

**Arguments**

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

**Examples**

```
getDensityPlot()
```

---

`getDomains`*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`*getDown get 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`      *getDownloadSection*

---

**Description**

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

**Usage**

```
getDownloadSection(flag = FALSE, choices = NULL)
```

**Arguments**

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

**Value**

the panel for download section in the menu;

**Note**

`getDownloadSection`

**Examples**

```
x<- getDownloadSection()
```

---

`getEnrichDO`                    *getEnrichDO*

---

**Description**

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

**Usage**

```
getEnrichDO(genelist = NULL, pvalueCutoff = 0.01)
```

**Arguments**

`genelist`,                gene list  
`pvalueCutoff`,          the p value cutoff

**Value**

enriched DO

**Note**

```
getEnrichD0
```

**Examples**

```
x <- getEnrichD0()
```

---

```
getEnrichG0
```

```
getEnrichGO
```

---

**Description**

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

**Usage**

```
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

**Note**

```
getEnrichGO
```

**Examples**

```
x <- getEnrichGO()
```

---

getEnrichKEGG	<i>getEnrichKEGG</i>
---------------	----------------------

---

**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()
```

---

getEntrezIds	<i>getGeneList</i>
--------------	--------------------

---

**Description**

Gathers the gene list to use for GOTerm analysis.

**Usage**

```
getEntrezIds(genes = NULL, org = "org.Hs.eg.db")
```

**Arguments**

genes,	gene list with fold changes
org,	organism for gene symbol entrez ID conversion

**Value**

ENTREZ ID list

**Note**

GOTerm  
getEntrezIds symbol to ENTREZ ID conversion

**Examples**

```
x <- getEntrezIds()
```

---

getGeneList

*getGeneList*

---

**Description**

Gathers the gene list to use for GOTerm analysis.

**Usage**

```
getGeneList(genes = NULL, org = "org.Hs.eg.db")
```

**Arguments**

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

**Value**

ENTREZ ID list

**Note**

GOTerm  
getGeneList symbol to ENTREZ ID conversion

**Examples**

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



---

<code>getGeneSetData</code>	<i>getGeneSetData</i>
-----------------------------	-----------------------

---

**Description**

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

**Usage**

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

**Arguments**

<code>data</code> ,	loaded dataset
<code>geneset</code> ,	given gene set

**Value**

data

**Examples**

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

---

<code>getGOLeftMenu</code>	<i>getGOLeftMenu</i>
----------------------------	----------------------

---

**Description**

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

**Usage**

```
getGOLeftMenu()
```

**Value**

returns the left menu according to the selected tab;

**Note**

```
getGOLeftMenu
```

**Examples**

```
x <- getGOLeftMenu()
```

getGoPanel                      *getGoPanel*

---

**Description**

Creates go term analysis panel within the shiny display.

**Usage**

```
getGoPanel()
```

**Value**

the panel for go term analysis;

**Note**

getGoPanel

**Examples**

```
x <- getGoPanel()
```

---

getGOPlots                      *getGOPlots*

---

**Description**

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

**Usage**

```
getGOPlots(dataset = NULL, input = NULL)
```

**Arguments**

dataset,	the dataset used
input,	input params

**Value**

the panel for go plots;

**Note**

getGOPlots

**Examples**

```
x<- getGOPlots()
```

---

getHelpButton	<i>getHelpButton prepares a helpbutton for to go to a specific site in the documentation</i>
---------------	--

---

**Description**

getHelpButton prepares a helpbutton for to go to a specific site in the documentation

**Usage**

```
getHelpButton(name = NULL, link = NULL)
```

**Arguments**

name,	name that are going to come after info
link,	link of the help

**Value**

the info button

**Examples**

```
x<- getHelpButton()
```

---

getHoverPlots	<i>getHoverPlots</i>
---------------	----------------------

---

**Description**

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

**Usage**

```
getHoverPlots(bardata = NULL, genename = NULL)
```

**Arguments**

bardata,	barplot data
genename,	gene name in the barplots

**Examples**

```
getHoverPlots()
```

---

<code>getInitialMenu</code>	<i>getInitialMenu</i>
-----------------------------	-----------------------

---

**Description**

Displays the initial menu within DEBrowser.

**Usage**

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

**Arguments**

<code>input,</code>	input from user
<code>output,</code>	output to user
<code>session,</code>	session info

**Value**

returns the initial menu

**Note**

`getInitialMenu`

**Examples**

```
x <- getInitialMenu()
```

---

<code>getIntHeatmap</code>	<i>getIntHeatmap</i>
----------------------------	----------------------

---

**Description**

`getIntHeatmap`

**Usage**

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

**Arguments**

<code>data,</code>	heatData
<code>input,</code>	all input params
<code>inputQCPlot,</code>	input poarams for QC

**Value**

plot

**Examples**

```
getIntHeatmap()
```

---

*getIntHeatmapVis*      *getIntHeatmapVis*

---

**Description**

Gathers the conditional panel for interactive heatmap

**Usage**

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

**Arguments**

randstr,      randstr

**Value**

the panel interactive heatmap

**Note**

`getIntHeatmapVis`

**Examples**

```
x <- getIntHeatmapVis()
```

---

*getIQRPlot*      *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()
```

---

getJsonObj	<i>getJsonObj</i>
------------	-------------------

---

**Description**

getJsonVars

**Usage**

```
getJsonObj(session = NULL, input = NULL, access_token = NULL)
```

**Arguments**

session,	session
input,	input
access_token,	access_token

**Examples**

```
x <- getJsonObj()
```

---

getKEGGModal	<i>getKEGGModal prepares a helpbutton for to go to a specific site in the documentation</i>
--------------	---

---

**Description**

getKEGGModal prepares a helpbutton for to go to a specific site in the documentation

**Usage**

```
getKEGGModal()
```

**Value**

the info button

**Examples**

```
x<- getKEGGModal()
```

---

getLeftMenu	<i>getLeftMenu</i>
-------------	--------------------

---

**Description**

Generates the left menu for for plots within the DEBrowser.

**Usage**

```
getLeftMenu(input = NULL)
```

**Arguments**

input,           input values

**Value**

returns the left menu according to the selected tab;

**Note**

getLeftMenu

**Examples**

```
x <- getLeftMenu()
```

---

getLegendSelect	<i>getLegendSelect</i>
-----------------	------------------------

---

**Description**

select legend

**Usage**

```
getLegendSelect()
```

**Note**

getLegendSelect

**Examples**

```
x <- getLegendSelect()
```

getLoadingMsg            *getLoadingMsg*

---

**Description**

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

**Usage**

```
getLoadingMsg(output = NULL)
```

**Arguments**

output,            output message

**Value**

loading msg

**Note**

getLoadingMsg

**Examples**

```
x <- getLoadingMsg()
```

---

getLogo                *getLogo*

---

**Description**

Generates and displays the logo to be shown within DEBrowser.

**Usage**

```
getLogo()
```

**Value**

return logo

**Note**

getLogo

**Examples**

```
x <- getLogo()
```



---

getMainPanel	<i>getMainPanel</i>
--------------	---------------------

---

**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	<i>getMainPanelPlots</i>
-------------------	--------------------------

---

**Description**

Gathers the the plots to be used within the main panel.

**Usage**

```
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**

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

---

```
getMainPlotsLeftMenu    getMainPlotsLeftMenu
```

---

**Description**

Generates the Main Plots Left menu to be displayed within the DEBrowser.

**Usage**

```
getMainPlotsLeftMenu()
```

**Value**

returns the left menu according to the selected tab;

**Note**

```
getMainPlotsLeftMenu
```

**Examples**

```
x <- getMainPlotsLeftMenu()
```

---

```
getMean                    getMean
```

---

**Description**

Gathers the mean for selected condition.

**Usage**

```
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

**Value**

data

**Examples**

```
x <- getMean()
```

---

getMergedComparison	<i>getMergedComparison</i>
---------------------	----------------------------

---

**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	<i>get the detail boxes after DE method selected</i>
------------------	--

---

**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**

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

---

```
getMostVariedList      getMostVariedList
```

---

**Description**

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

**Usage**

```
getMostVariedList(datavar = NULL, cols = NULL, input = NULL)
```

**Arguments**

datavar,	loaded dataset
cols,	selected columns
input,	input

**Value**

data

**Examples**

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

---

```
getNormalizedMatrix      getNormalizedMatrix
```

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

normalized matrix

**Note**

getGoPanel

**Examples**

```
x <- getNormalizedMatrix(mtcars)
```

---

getOrganism	<i>getOrganism</i>
-------------	--------------------

---

**Description**

getOrganism

**Usage**

```
getOrganism(org)
```

**Arguments**

org,                    organism

**Value**

organism name for keg

**Note**

getOrganism

**Examples**

```
x <- getOrganism()
```

---

getOrganismBox	<i>getOrganismBox</i>
----------------	-----------------------

---

**Description**

Get the organism Box.

**Usage**

```
getOrganismBox()
```

**Value**

selectInput

**Note**

`getOrganismBox`  
`getOrganismBox` makes the organism box

**Examples**

```
x <- getOrganismBox()
```

---

`getOrganismPathway`      *getOrganismPathway*

---

**Description**

`getOrganismPathway`

**Usage**

```
getOrganismPathway(org)
```

**Arguments**

`org`,                      `organism`

**Value**

organism name for pathway

**Note**

`getOrganismPathway`

**Examples**

```
x <- getOrganismPathway()
```

---

`getPCAexplained`              *getPCAexplained*

---

**Description**

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

**Usage**

```
getPCAexplained(datasetInput = NULL, input = NULL)
```

**Arguments**

datasetInput, selected data  
input, from user

**Value**

explained plot

**Examples**

```
load(system.file("extdata", "demo", "demodata.Rda",  
package="debrowser"))  
input<-c()  
input$qcplot<-"pca"  
input$col_list<-colnames(demodata[,2:7])  
x <- getPCAexplained(getNormalizedMatrix(demodata[,2:7]),  
input)
```

---

*getPCselection**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()
```

---

<code>getProgramTitle</code>	<i>getProgramTitle</i>
------------------------------	------------------------

---

**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()
```

---

<code>getQCLeftMenu</code>	<i>getQCLeftMenu</i>
----------------------------	----------------------

---

**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
```



**Examples**

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

---

<i>getQCPanel</i>	<i>getQCPanel</i>
-------------------	-------------------

---

**Description**

Gathers the conditional panel for QC plots

**Usage**

```
getQCPanel(input = NULL)
```

**Arguments**

input,            user input

**Value**

the panel for QC plots

**Note**

*getQCSection*

**Examples**

```
x <- getQCPanel()
```

---

<i>getQCPlots</i>	<i>getQCPlots</i>
-------------------	-------------------

---

**Description**

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

**Usage**

```
getQCPlots(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

**Value**

the panel for QC plots

**Note**

getQCPlots

**Examples**

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

---

getQCReplot

*getQCReplot*

---

**Description**

Prepares QCplots for comparisons and others

**Usage**

```
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**

```
x <- getQCReplot()
```

---

getSampleNames	<i>getSampleNames</i>
----------------	-----------------------

---

**Description**

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

**Usage**

```
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**

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

---

getSamples	<i>getSamples</i>
------------	-------------------

---

**Description**

Gathers the sample names to be used within DEBrower.

**Usage**

```
getSamples(cnames = NULL, index = 1)
```

**Arguments**

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

**Value**

choices

**Examples**

```
x <- getSamples()
```

---

<code>getSearchData</code>	<i>getSearchData</i>
----------------------------	----------------------

---

**Description**

search the geneset in the tables and return it

**Usage**

```
getSearchData(dat = NULL, input = NULL)
```

**Arguments**

<code>dat</code> ,	table data
<code>input</code> ,	input params

**Value**

data

**Examples**

```
x <- getSearchData()
```

---

<code>getSelectedCols</code>	<i>getSelectedCols</i>
------------------------------	------------------------

---

**Description**

gets selected columns

**Usage**

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

**Arguments**

<code>data</code> ,	all loaded data
<code>datasetInput</code> ,	selected dataset
<code>input</code> ,	user input params

**Examples**

```
getSelectedCols()
```

---

getSelectedDatasetInput  
*getSelectedDatasetInput*

---

**Description**

Gathers the user selected dataset output to be displayed.

**Usage**

```
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**

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

---

getSelectInputBox      *getSelectInputBox*

---

**Description**

Selects user input conditions to run in DESeq.

**Usage**

```
getSelectInputBox(id = NULL, name = NULL, num = 0, choices = NULL,  
  selected = NULL, cw = 2)
```

**Arguments**

<code>id,</code>	input id
<code>name,</code>	label of the box
<code>num,</code>	panel that is going to be shown
<code>choices,</code>	sample list
<code>selected,</code>	selected sample list
<code>cw,</code>	column width

**Examples**

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

---

`getSelHeat`

*getSelHeat*

---

**Description**

heatmap selection functionality

**Usage**

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

**Arguments**

<code>data,</code>	selected genes
<code>input,</code>	input params

**Value**

plot

**Examples**

```
x <- getSelHeat()
```

---

getShapeColor	<i>getShapeColor</i>
---------------	----------------------

---

**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**

```
x <- getShapeColor()
```

---

getStartPlotsMsg	<i>getStartPlotsMsg</i>
------------------	-------------------------

---

**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**

```
x <- getStartPlotsMsg()
```

---

getStartupMsg	<i>getStartupMsg</i>
---------------	----------------------

---

**Description**

Generates and displays the starting message within DEBrowser.

**Usage**

```
getStartupMsg()
```

**Value**

return startup msg

**Note**

getStartupMsg

**Examples**

```
x <- getStartupMsg()
```

---

getTableStyle	<i>getTableStyle</i>
---------------	----------------------

---

**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()
```



---

getTextOnOff                      *getTextOnOff*

---

**Description**

text on PCA plot on and off

**Usage**

```
getTextOnOff()
```

**Note**

```
getTextOnOff
```

**Examples**

```
x <- getTextOnOff()
```

---

getToolTipPCA                      *getToolTipPCA*

---

**Description**

Prepares tooltip text for PCA plot

**Usage**

```
getToolTipPCA(dat = NULL)
```

**Arguments**

```
dat,                      data
```

**Value**

tooltip text

**Examples**

```
x <- getToolTipPCA()
```

---

getToolTipText	<i>getToolTipText</i>
----------------	-----------------------

---

**Description**

Prepares tooltip text 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	<i>getUp get up regulated data</i>
-------	------------------------------------

---

**Description**

getUp get up regulated data

**Usage**

```
getUp(filt_data = NULL)
```

**Arguments**

filt\_data, filt\_data

**Value**

data

**Examples**

```
x <- getUp()
```

---

getUpDown	<i>getUpDown get up+down regulated data</i>
-----------	---

---

**Description**

getUpDown get up+down regulated data

**Usage**

```
getUpDown(filt_data = NULL)
```

**Arguments**

filt\_data,      filt\_data

**Value**

data

**Examples**

```
x <- getUpDown()
```

---

get_state_id	<i>get_state_id</i>
--------------	---------------------

---

**Description**

Helper to copy the bookmark to a user named directory

**Usage**

```
get_state_id(prev_url = NULL)
```

**Arguments**

prev\_url,      prev url

**Examples**

```
x <- get_state_id()
```

---

get_user_info	<i>Get the logged in user's email and other info</i>
---------------	--

---

**Description**

Get the logged in user's email and other info

**Usage**

```
get_user_info(id = "me")
```

**Arguments**

id ID of the person to get the profile data for. 'me' to get current user.

**Value**

A People resource

<https://developers.google.com/+/web/api/rest/latest/people#resource-representations>

**See Also**

<https://developers.google.com/+/web/api/rest/latest/people>

**Examples**

```
## Not run:
options(googleAuthR.scopes.selected =
  c("https://www.googleapis.com/auth/userinfo.email",
    "https://www.googleapis.com/auth/userinfo.profile"))

googleAuthR::gar_auth()

## default is user logged in
user <- get_user_info()

## End(Not run)
```

---

hideObj	<i>hideObj</i>
---------	----------------

---

**Description**

Hides a shiny object.

**Usage**

```
hideObj(btns = NULL)
```

**Arguments**

btns,                   hide group of objects with shinyjs

**Examples**

```
x <- hideObj()
```

---

installpack	<i>installpack</i>
-------------	--------------------

---

**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	<i>link_brush</i>
------------	-------------------

---

**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()
```

---

loadpack	<i>loadpack</i>
----------	-----------------

---

**Description**

load packages

**Usage**

```
loadpack(package_name = NULL)
```

**Arguments**

package\_name, package name to be loaded

**Note**

loadpack

**Examples**

```
x <- loadpack()
```

---

load_data	<i>load_data.</i>
-----------	-------------------

---

**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 ()
```

---

logSliderJScore	<i>logSliderJScore</i>
-----------------	------------------------

---

**Description**

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

**Usage**

```
logSliderJScore(slidername = NULL)
```

**Arguments**

slidername,      id of the slider

**Value**

returns the slider values in log10 scale

**Note**

logSliderJScore

**Examples**

```
x <- logSliderJScore()
```

---

mainScatter	<i>mainScatter</i>
-------------	--------------------

---

**Description**

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

**Usage**

```
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**

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

---

MAPlot

*MAPlot*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

MA plot

**Examples**

```
x <- MAPlot()
```



MAZoom

*MAZoom***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**

<code>dat</code> ,	dataframe that has log2FoldChange and log10padj values
<code>data_tooltip</code> ,	tooltip specific to this plot
<code>domains</code> ,	the domains to be colored
<code>colors</code> ,	colors for each domain

**Value**

zoomed MA plot

**Examples**

```
x <- MAZoom()
```

panel.cor

*panel.cor***Description**

Prepares the correlations for the all2all plot.

**Usage**

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

**Arguments**

<code>x</code> ,	numeric vector x
<code>y</code> ,	numeric vector y
<code>prefix</code> ,	prefix for the text
<code>cex.cor</code> ,	correlation font size
<code>...</code> ,	additional parameters

**Value**

all2all correlation plots

**Examples**

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

---

panel.hist

*panel.hist*

---

**Description**

Prepares the histogram for the all2all plot.

**Usage**

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

**Arguments**

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

**Value**

all2all histogram plots

**Examples**

```
panel.hist(1)
```

---

plot\_pca

*plot\_pca*

---

**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")
```

---

```
prepAddQCPlots
```

```
prepAddQCPlots
```

---

**Description**

prepares IQR and density plots

**Usage**

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

**Arguments**

data,	barplot data
input,	user input params

**Examples**

```
prepAddQCPlots()
```

---

prepDataContainer      *prepDataContainer*

---

**Description**

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()
```

---

prepDataForQC      *prepDataForQC*

---

**Description**

Prepares selected data for QC plots.

**Usage**

```
prepDataForQC(dataset = NULL, input = NULL)
```

**Arguments**

dataset,	loaded dataset
input,	input

**Value**

data

**Examples**

```
x <- prepDataForQC()
```

---

```
prepDEOutput      prepDEOutput
```

---

**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, input = NULL)
```

**Arguments**

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

**Value**

data

**Examples**

```
x <- prepDEOutput()
```

---

```
push      push
```

---

**Description**

Push an object to the list.

**Usage**

```
push(l, ...)
```

**Arguments**

l,	that are going to push to the list
...,	list object

**Value**

combined list

**Examples**

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

---

readMetaData	<i>readMetaData</i>
--------------	---------------------

---

**Description**

read metadata file

**Usage**

```
readMetaData(input = NULL)
```

**Arguments**

input,                   input values

**Note**

readMetaData

**Examples**

```
x <- readMetaData()
```

---

removeBookmark	<i>removeBookmark</i>
----------------	-----------------------

---

**Description**

remove saved state

**Usage**

```
removeBookmark(ID = NULL, username = NULL)
```

**Arguments**

ID,                   prev url  
username,            username

**Examples**

```
x <- removeBookmark()
```

---

removeCols	<i>removeCols</i>
------------	-------------------

---

**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	<i>round_vals</i>
------------	-------------------

---

**Description**

Plot PCA results.

**Usage**

```
round_vals(1)
```

**Arguments**

1,	the value
----	-----------

**Value**

round value

**Examples**

```
x<-round_vals(5.1323223)
```

---

runBayseq	<i>runBayseq</i>
-----------	------------------

---

**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	<i>runDE</i>
-------	--------------

---

**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**

de results

**Examples**

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

---

runDESeq

*runDESeq*

---

**Description**

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

**Usage**

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

**Arguments**

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

**Value**

deseq2 results

**Examples**

```
x <- runDESeq(data<-NULL, columns<-c())
```

---

runDESeq2

*runDESeq2*


---

### Description

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

### Usage

```
runDESeq2(data = NULL, columns = NULL, conds = NULL,
          fitType = c("parametric", "local", "mean"), betaPrior = 0,
          testType = c("Wald", "LRT"), rowsum.filter = 10)
```

### Arguments

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

### Value

deseq2 results

### Examples

```
x <- runDESeq2()
```

runEdgeR

*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**

<code>data</code> ,	A matrix that includes all the expression raw counts, rownames has to be the gene, isoform or region names/IDs
<code>columns</code> ,	is a vector that includes the columns that are going to be analyzed. These columns has to match with the given data.
<code>conds</code> ,	experimental conditions. The order has to match with the column order
<code>normfact</code> ,	Calculate normalization factors to scale the raw library sizes. Values can be "TMM", "RLE", "upperquartile", "none".
<code>dispersion</code> ,	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.
<code>testType</code> ,	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.
<code>rowsum.filter</code> ,	regions/genes/isoforms with total count (across all samples) below this value will be filtered out

**Value**

edgeR results

**Examples**

```
x <- runEdgeR()
```

---

runHeatmap	<i>runHeatmap</i>
------------	-------------------

---

### 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	<i>runLimma</i>
----------	-----------------

---

### 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,	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".
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**

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

---

run\_pca

*run\_pca*

---

**Description**

Runs PCA on the selected dataset.

**Usage**

```
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

**Examples**

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

---

`saveQCPlot`*saveQCPlot*

---

**Description**

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

**Usage**

```
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**

```
saveQCPlot()
```

---

scatterZoom	<i>scatterZoom</i>
-------------	--------------------

---

**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	<i>selectBatchEffect</i>
-------------------	--------------------------

---

**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

**Note**

```
selectBatchEffect
```

**Examples**

```
x <- selectBatchEffect()
```

---

selectConditions	<i>selectConditions</i>
------------------	-------------------------

---

**Description**

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

**Usage**

```
selectConditions(Dataset = NULL, choicecounter, input = NULL,  
loadingJSON = NULL)
```

**Arguments**

Dataset,	used dataset
choicecounter,	total number of comparisons
input,	input params
loadingJSON,	loads from json

**Value**

the panel for go plots;

**Note**

```
selectConditions
```

**Examples**

```
x<- selectConditions()
```



---

selectedInput	<i>selectedInput</i>
---------------	----------------------

---

**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	<i>setFilterParams</i>
-----------------	------------------------

---

**Description**

It sets the filter parameters

**Usage**

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

**Arguments**

session,	session variable
input,	input parameters

**Examples**

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

---

showObj	<i>showObj</i>
---------	----------------

---

**Description**

Displays a shiny object.

**Usage**

```
showObj(btns = NULL)
```

**Arguments**

btns, show group of objects with shinyjs

**Examples**

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

---

startDEBrowser	<i>startDEBrowser</i>
----------------	-----------------------

---

**Description**

Starts the DEBrowser to be able to run interactively.

**Usage**

```
startDEBrowser()
```

**Value**

the app

**Note**

```
startDEBrowser
```

**Examples**

```
startDEBrowser()
```

textareaInput                      *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                      *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

*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

*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()
```

---

whitelist

*Whitelist check*

---

**Description**

After a user logs in, check to see if they are on a whitelist

**Usage**

```
whitelist(user_info, whitelist = NULL)
```

**Arguments**

`user_info` the object returned by [get\\_user\\_info](#)  
`whitelist` A character vector of emails on whitelist

**Value**

TRUE if on whitelist or no whitelist, FALSE if not

**Examples**

```
## Not run:
options(googleAuthR.scopes.selected =
  c("https://www.googleapis.com/auth/userinfo.email",
    "https://www.googleapis.com/auth/userinfo.profile"))

googleAuthR::gar_auth()

## default is user logged in
user <- get_user_info()

the_list <- whitelist(user, c("your@email.com",
  "another@email.com",
  "yet@anotheremail.com"))

if(the_list){
  message("You are on the list.")
} else {
  message("If you're not on the list, you're not getting in.")
}

## End(Not run)
```

# Index

actionButton, 4  
add\_title\_pos, 6  
addDataCols, 5  
addID, 6  
all2all, 7  
applyFilters, 7  
applyFiltersToMergedComparison, 8  
  
bookmarkServer, 9  
bookmarkUI, 9  
  
clusterData, 10  
compareClust, 10  
copy2newDirectory, 11  
correctBatchEffect, 11  
  
deServer, 12  
deUI, 13  
drawPCAExplained, 13  
  
get\_state\_id, 51  
get\_user\_info, 52, 77  
getAfterLoadMsg, 14  
getColors, 14  
getColorShapeSelection, 15  
getCompSelection, 15  
getConditionSelector, 16  
getConditionSelectorFromMeta, 16  
getCondMsg, 17  
getCutOffSelection, 17  
getDataForTables, 18  
getDataPrepPanel, 19  
getDensityPlot, 19  
getDomains, 20  
getDown, 20  
getDownloadSection, 21  
getEnrichDO, 21  
getEnrichGO, 22  
getEnrichKEGG, 23  
getEntrezIds, 23  
getGeneList, 24  
getGeneSetData, 25  
getGOLeftMenu, 25  
getGoPanel, 26  
getGOPlots, 26  
getHelpButton, 27  
getHoverPlots, 27  
getInitialMenu, 28  
getIntHeatmap, 28  
getIntHeatmapVis, 29  
getIQRPlot, 29  
getJsonObj, 30  
getKEGGModal, 30  
getLeftMenu, 31  
getLegendSelect, 31  
getLoadingMsg, 32  
getLogo, 32  
getMainPanel, 33  
getMainPanelPlots, 33  
getMainPlotsLeftMenu, 34  
getMean, 34  
getMergedComparison, 35  
getMethodDetails, 35  
getMostVariedList, 36  
getNormalizedMatrix, 36  
getOrganism, 37  
getOrganismBox, 37  
getOrganismPathway, 38  
getPCAExplained, 38  
getPCselection, 39  
getProgramTitle, 40  
getQCLeftMenu, 40  
getQCPanel, 41  
getQCPlots, 41  
getQCReplot, 42  
getSampleNames, 43  
getSamples, 43  
getSearchData, 44  
getSelectedCols, 44  
getSelectedDatasetInput, 45  
getSelectInputBox, 45  
getSelHeat, 46  
getShapeColor, 47  
getStartPlotsMsg, 47  
getStartupMsg, 48  
getTableStyle, 48  
getTextOnOff, 49

getToolTipPCA, 49  
getToolTipText, 50  
getUp, 50  
getUpDown, 51  
  
hideObj, 52  
  
installpack, 53  
  
link\_brush, 53  
load\_data, 54  
loadpack, 54  
logSliderJScore, 55  
  
mainScatter, 55  
MAPlot, 56  
MAZoom, 57  
  
panel.cor, 57  
panel.hist, 58  
plot\_pca, 58  
prepAddQCPlots, 59  
prepDataContainer, 60  
prepDataForQC, 60  
prepDEOutput, 61  
push, 61  
  
readMetaData, 62  
removeBookmark, 62  
removeCols, 63  
round\_vals, 63  
run\_pca, 69  
runBayseq, 64  
runDE, 64  
runDESeq, 65  
runDESeq2, 66  
runEdgeR, 67  
runHeatmap, 68  
runLimma, 68  
  
saveQCPlot, 70  
scatterZoom, 71  
selectBatchEffect, 71  
selectConditions, 72  
selectedInput, 73  
setFilterParams, 73  
showObj, 74  
startDEBrowser, 74  
  
textareaInput, 75  
togglePanels, 75  
  
volcanoPlot, 76  
volcanoZoom, 76  
  
whitelist, 77