

# Package ‘ReportingTools’

July 2, 2025

**Title** Tools for making reports in various formats

**Version** 2.49.0

**Author** Jason A. Hackney, Melanie Huntley, Jessica L. Larson, Christina Chaivorapol, Gabriel Becker, and Josh Kaminker

**Maintainer** Jason A. Hackney <hackney.jason@gene.com>, Gabriel Becker <becker.gabe@gene.com>, Jessica L. Larson <larson.jessica@gmail.com>

**Depends** methods, knitr, utils

**Imports** Biobase, hwriter, Category, GOstats, limma(>= 3.17.5), lattice, AnnotationDbi, edgeR, annotate, PFAM.db, GSEABase, BiocGenerics(>= 0.1.6), grid, XML, R.utils, DESeq2(>= 1.3.41), ggplot2, ggbio, IRanges

**Suggests** RUnit, ALL, hgu95av2.db, org.Mm.eg.db, shiny, pasilla, org.Sc.sgd.db, rmarkdown, markdown

**Type** Package

**LazyLoad** yes

**License** Artistic-2.0

**Description** The ReportingTools software package enables users to easily display reports of analysis results generated from sources such as microarray and sequencing data. The package allows users to create HTML pages that may be viewed on a web browser such as Safari, or in other formats readable by programs such as Excel. Users can generate tables with sortable and filterable columns, make and display plots, and link table entries to other data sources such as NCBI or larger plots within the HTML page. Using the package, users can also produce a table of contents page to link various reports together for a particular project that can be viewed in a web browser. For more examples, please visit our site: <http://research-pub.gene.com/ReportingTools>.

**ByteCompile** TRUE

**VignetteBuilder** utils, rmarkdown

**biocViews** ImmunoOncology, Software, Visualization, Microarray, RNASeq, GO, DataRepresentation, GeneSetEnrichment

**git\_url** <https://git.bioconductor.org/packages/ReportingTools>

**git\_branch** devel

**git\_last\_commit** 5605228  
**git\_last\_commit\_date** 2025-04-15  
**Repository** Bioconductor 3.22  
**Date/Publication** 2025-07-01

**Contents**

BaseReport-class . . . . .	2
CSVFile . . . . .	3
CSVFile-class . . . . .	4
custHeaderPanel . . . . .	5
DataPackage . . . . .	5
DataPackage-class . . . . .	6
filename-methods . . . . .	7
finish-methods . . . . .	8
HTMLReport-class . . . . .	8
HTMLReportRef-class . . . . .	9
Link . . . . .	10
makeOldHTMLReport . . . . .	11
mockRnaSeqData . . . . .	12
modifyReportDF-methods . . . . .	13
objectToHTML-methods . . . . .	13
publish-methods . . . . .	14
readReport . . . . .	16
ReportHandlers-class . . . . .	17
reporting.theme . . . . .	18
reporting.theme.alternate . . . . .	19
toReportDF-methods . . . . .	19
validConnection . . . . .	20
<b>Index</b>	<b>21</b>

---

BaseReport-class	<i>Class "BaseReport"</i>
------------------	---------------------------

---

**Description**

A ReportType defines a way of representing an R object in a different, ordered format. Several ReportTypes have been detailed in this package, including classes for HTML reports and R data packages.

**Objects from the Class**

Objects can be created by calls of the form `new("BaseReport", ...)`. This is an abstract class that is inherited by other ReportTypes, and should not be used directly.

**Slots**

- shortName:** A character string generally used as the filename for reports
- title:** A character string that gives a longer description of what the report shows.
- reportDirectory:** A file path that details where the report will be saved.

**Methods**

**name** signature(object = "BaseReport"): Get the shortName assigned to the report.

**reportDirectory** signature(object = "BaseReport"): Get the directory where the report will be generated.

**reportDirectory<-** signature(object = "BaseReport", value = "character"): Set the directory where the report will be generated.

**show** signature(object = "BaseReport"): ...

**title** signature(main = "BaseReport"): Return the title of the report

**title<-** signature(object = "BaseReport", value = "character"): Set the title of the report.

**path** signature(object = "BaseReport"): Get the filesystem location of the report.

**See Also**

[HTMLReport DataPackage CSVFile](#)

**Examples**

```
showClass("BaseReport")
```

---

CSVFile

*Function for creating and initializing a CSVFile for publishing results*

---

**Description**

A CSVFile is a pointer to a comma separated value file on the file system. Publishing to a CSVFile overwrites the current contents of the given CSVFile.

**Usage**

```
CSVFile(shortName, title = "", reportDirectory = ".")
```

**Arguments**

shortName	A character string giving a short title for the report. Used as the base of the filename.
reportDirectory	A character string giving the location of the report.
title	A character string giving a longer description of the report.

**Value**

An object of class CSVFile, which will represents a file to be written to. There are no open filehandles, and calling publish multiple times will result in the file being overwritten.

**See Also**

[CSVFile-class](#), [HTMLReport-class](#)

## Examples

```
my.df <- data.frame(EGID = c("103", "104", "105", "106", "107"),
                    RPKM = c(4, 5, 3, 100, 75),
                    DE = c("Yes", "Yes", "No", "No", "No"))
csv.file <- CSVFile(shortName = "my_csv_file",
                    reportDirectory = "reportDirectory")
publish(my.df, csv.file)
```

---

CSVFile-class	<i>Class "CSVFile"</i>
---------------	------------------------

---

## Description

A CSVFile is a pointer to a comma separated value file on the file system. Publishing to a CSVFile overwrites the current contents of the given CSVFile.

## Objects from the Class

Objects from this class...

## Slots

**shortName:** A character string used as the base of the filename. '.csv' is appended to the shortName to make the full file name.

**title:** The title of the report. This doesn't appear anywhere in the output, but can be used to track the report in an R session.

**reportDirectory:** A file path that details where the csv file will be saved.

## Extends

Class "[BaseReport](#)", directly.

## Methods

**filename** The name of the csv file, including extension

**name<-** Set the name of the CSVFile object

## Examples

```
showClass("CSVFile")
```

---

<code>custHeaderPanel</code>	<i>Define a header panel with custom JS/CSS for a Shiny app</i>
------------------------------	-----------------------------------------------------------------

---

**Description**

Defines a header panel which loads custom Javascript and CSS files.

**Usage**

```
custHeaderPanel(title, windowTitle = title, js = NULL, css = NULL)
```

**Arguments**

<code>title</code>	Title of the shiny app
<code>windowTitle</code>	Title of the window for the shiny app
<code>js</code>	Custom javascript to be loaded in the app
<code>css</code>	Custom CSS to be loaded in the app

**Details**

See the shiny vignette for example usage.

**Value**

The return value is not directly meaningful to the end user and is intended to be passed into shiny layout functions such as `pageWithSidebar`.

**Author(s)**

Gabriel Becker and Jessica L. Larson

**See Also**

[pageWithSidebar](#)

---

<code>DataPackage</code>	<i>Function for creating and initializing an object of class <code>DataPackage</code>.</i>
--------------------------	--------------------------------------------------------------------------------------------

---

**Description**

This is a function for creating a `DataPackage`. Calls to publish an object to the package serialize the object to disk, adding a file to the data directory, and adding a stubbed `.Rd` documentation file for the dataset.

**Usage**

```
DataPackage(shortName, title = "", reportDirectory = ".", version =  
"0.0.1", dependencies = c("Biobase"), license = "", description = "",  
author = "nobody", maintainer = "nobody <nobody@nothing.net>")
```

**Arguments**

shortName	A character string giving the title of the DataPackage. This is used as both the title in the DESCRIPTION file and as the directory name for the package.
reportDirectory	Where the DataPackage directory will be created.
title	A character string giving the title of the package.
version	A character string giving the version of the DataPackage.
dependencies	A character vector listing what packages the DataPackage depends on.
license	A character string detailing the license the DataPackage published under.
description	A character string giving the description of the DataPackage.
author	A character string giving the author of the package.
maintainer	A character string giving the maintainer of the package.

**Value**

An object of class DataPackage. As a side effect, the directory structure of the data package will also be created in the location given by reportDirectory. On publication, objects are saved to the data directory of the package, and whatever dependencies the objects imply are added to the list of package dependencies.

**See Also**

[DataPackage-class](#)

**Examples**

```
my.df <- data.frame(EGID = c("103", "104", "105", "106", "107"),
                    RPKM = c(4, 5, 3, 100, 75),
                    DE = c("Yes", "Yes", "No", "No", "No"))
data.package <- DataPackage('MyPackage', title = "My awesome package",
                           author = "J.J. Nobody", maintainer = "J.J. Nobody <nobody@nowhere.net>")
publish('my.df', data.package)
```

---

DataPackage-class	<i>Class "DataPackage"</i>
-------------------	----------------------------

---

**Description**

This is a pointer to an R data package. The result should be an installable R data package, with documentation for the data objects published therein.

**Objects from the Class**

Objects can be created by calls of the form `new("DataPackage", ...)`.

## Slots

**version:** The current version of the generated data package  
**dependencies:** A character vector of packages that the data package depends on.  
**author:** The author(s) of the data package.  
**maintainer:** The maintainer(s) of the data package. Defaults to the same person as the author.  
**license:** What license to use when creating the data package.  
**description:** The description of the data package.  
**package.Rd:** Rd string for package-level help for the generated data package.  
**shortName:** Name of the data package.  
**title:** The title of the package  
**reportDirectory:** The directory in which the data package is generated.

## Extends

Class "[BaseReport](#)", directly.

## Methods

**dependencies** Retrieve the list of dependencies for the data package.  
**dependencies<-** Set the list of dependencies for the data package.  
**finish** This is the final step in publishing objects to the data package. It recreates the DESCRIPTION file for the data package. The package name, title, version, author, maintainer, dependencies, license and description are set from the appropriate slots in the DataPackage object.

## Examples

```

my.df <- data.frame(EGID = c("103", "104", "105", "106", "107"),
                    RPKM = c(4, 5, 3, 100, 75),
                    DE = c("Yes", "Yes", "No", "No", "No"))
data.package <- DataPackage('MyPackage', title = "My awesome package",
                             author = "J.J. Nobody", maintainer = "J.J. Nobody <nobody@nowhere.net>")
publish('my.df', data.package)

dependencies(data.package) # Returns "Biobase"
dependencies(data.package) <- c('Biobase', 'GSEABase')

finish(data.package)
  
```

---

filename-methods

*Methods for getting the name of a file for a CSVFile or HTMLReport.*


---

## Description

These methods return the filename for a report, for report types where a single file is produced.

## Methods

**signature(object = "CSVFile")** Return the file name of the CSVFile. This is generated by adding '.csv' to the shortName of the report.  
**signature(object = "HTMLReport")** Return the file name of the HTMLReport. This is generated by adding '.html' to the shortName of the report.

---

finish-methods	<i>Finalizing reports after publishing results</i>
----------------	----------------------------------------------------

---

### Description

This is a method for finalizing a report after results have been published. The exact nature of finalizing depends on the report type, as detailed below.

### Methods

`signature(publicationType = "DataPackage")` Calling `finish` on a `DataPackage` object rewrites the DESCRIPTION file, making sure that all of the dependencies for the objects in the DataPackage are listed.

`signature(publicationType = "HTMLReport")` Calling `finish` on an `HTMLReport` calls the `hwrite` function `closePage`, which closes the body and html tags on the page and closes the connection to the file.

`signature(publicationType = "HTMLReportRef")` Calling `finish` on an `HTMLReportRef` calls all the finish handler for all `ReportHandlers` attached to the report. These handlers perform various actions depending on the desired behavior of the report.

### Examples

```
my.df <- data.frame()
## html.report <- HTMLReport(shortName = "my_html_file",
##   reportDirectory = "reportDirectory")
# publish(my.df, html.report)
## finish(html.report)
```

---

HTMLReport-class	<i>Class "HTMLReport"</i>
------------------	---------------------------

---

### Description

This class has been superceded by `HTMLReportRef` and is deprecated. The `HTMLReport` constructor function now creates an `HTMLReportRef` object.

### Objects from the Class

Objects can be created by calls of the form `new("HTMLReport", ...)`.

### Slots

**baseUrl:** What is the base URL to which the `HTMLReport`'s filename is appended.

**basePath:** What is the base file system URI to which the `HTMLReport`'s filename is appended.

**page:** The connection to the html file. See `openPage` in the `hwriter` package.

**shortName:** A character string used as the base of the filename. `'.html'` is appended to the shortName to make the full file name.

**title:** The title of the `HTMLReport`. This is used as the page title in the header, and is also printed on the page.

**reportDirectory:** A file path that details where the report will be saved.

**Extends**

Class "[BaseReport](#)", directly.

**Methods**

**basePath** Return the basePath for the HTMLReport

**baseUrl** Return the baseUrl for the HTMLReport

**filename** Return the file name of the HTMLReport. This is generated by adding '.html' to the shortName of the report.

**finish** This is the final step in publishing results to an HTMLReport. It prints the closing tags for the html body, and then closes the connection to the html file. Further calls to publish will fail because the connection is closed.

**name<-** Set the short name for the html file.

**page** Returns the connection to the html file.

**page<-** Set the connection to the html file.

**validConnection** Is the connection to the html file still open? If finish has been called, this will return FALSE.

---

HTMLReportRef-class	Class "HTMLReportRef"
---------------------	-----------------------

---

**Description**

A referenceClass-based representation of an HTML report generated using R objects. This class is based around a persistent, DOM-based representation of the HTML page being created, which is created during initialization and updated as the report is manipulated. HTMLReport is the constructor function for the class.

**Details**

Publication behavior of HTMLReportRef reports is controlled via an event-observer paradigm. Developers specify behavior in the form of one or more ReportHandlers objects. These objects contain instructions (in the form of R functions) to be carried out whenever certain actions are taken. These actions are: initial report creation ("init"), elements being added to the report ("addElement"), elements being removed from the report ("removeElement"), and the report being finalized ("finish"). See [ReportHandlers](#) for more detail.

**Methods**

**[[<-** signature(x = "HTMLReportRef", i = "ANY", j = "ANY", value = "ANY"): Place an element representing value into the report. This will replace any existing element by the same name in the report. x\$addElement is used to perform the actual report manipulation.

**[[** signature(x = "HTMLReportRef", i = "ANY"): Access an element within the report. The element will be returned in its DOM/HTML representation.

**publish** signature(object = "ANY", publicationType = "HTMLReportRef"): Place an element representing object into the report. publicationType\$addElement is used to perform the actual report manipulation.

**path** signature(object = "HTMLReportRef"): The paths returned by each of the ReportHandlers for the report.

**Fields**

**shortName:** Short version of report name.

**title:** Title of the report.

**reportDirectory:** Directory (relative to basePath) in which to publish the report.

**basePath:** Base path to publish to.

**handlers:** A list of ReportHandlers objects specifying actions to be taken in response specific events during the report creation/manipulation process. See Details.

**Class-Based Methods**

**prepare(obj, ...):** Prepare an R object to be added to the report by converting it to an HTML representation. Currently this is done by calling the objectToHTML function on the object, though this is likely to change in the future.

**initialize(...):** Initialize the report, including generating the base HTML structure. The init function for each ReportHandlers object in handlers is called at the end of this process.

**addElement(name, value, ...):** Add an element to the report. prepare is called to generate an HTML representation for value. That representation is then added to the internal DOM representation of the report. The addElement function for each ReportHandlers object in handlers is called at the end of this process.

**finish():** Finalize a report. This causes the finish function on each element of handlers to be called. This is something of a misnomer in that the report can be further manipulated and "finished" again after this function has been called.

**Author(s)**

Gabriel Becker <gmbecker@ucdavis.edu>

**See Also**

[ReportHandlers](#)

**Examples**

```
showClass("HTMLReportRef")
```

---

Link

*Create a table of Links*

---

**Description**

This is a convenience function for creating tables of links (eg for an index page).

**Usage**

```
Link(obj, target = NA, report = NULL)
```

**Arguments**

obj	An object to be turned into one or more links. Typically a character vector of link text or a list of HTMLReportRef objects
target	The link targets for each created link.
report	A report (HTMLReportRef object) whose location the generated links should be relative to.

**Details**

If target is NA, Link attempts to infer the link target(s) based on obj

**Value**

An XMLInternalNode corresponding to HTML for either a single link or a div containing multiple links separated each on separate lines

**Author(s)**

Gabriel Becker

**Examples**

```
mylinks = Link(c("Link1", "Link2"), c("www.google.com", "www.cnn.com"))
```

---

makeOldHTMLReport	<i>Creating and initializing an HTMLReport object</i>
-------------------	-------------------------------------------------------

---

**Description**

This is the function one should use to instantiate an HTMLReport. It basically creates the report in the appropriate place, creating directories if needed. Special care should be taken if non-default javascript or css files are used, as detailed below.

**Usage**

```
makeOldHTMLReport(shortName, title = NULL, reportDirectory = ".",
  baseUrl = "localhost", basePath = NULL, page = NULL, link.css = NULL,
  link.javascript = NULL, overwrite.js=TRUE)
```

**Arguments**

shortName	A short name for the report. It is used as the base of the HTML file.
title	The title for the page. It will be used in the title element in the page head, and will also be written at the top of the page.
reportDirectory	A character string giving the directory where the report will be written.
baseUrl	A character string giving the location of the page in terms of HTML locations.
basePath	A character string giving the location of the page in terms of filesystem locations.
page	The file handle for the html page, returned from the hwr i t e r function openPage.

<code>link.css</code>	A character vector of file URLs detailing where to get style sheets. If unset, the default styling for ReportingTools is used.
<code>link.javascript</code>	A character vector of file URLs detailing where to get javascript. If unset, the default javascript libraries for ReportingTools are used.
<code>overwrite.js</code>	logical whether css and javascript files should be overwritten when copied into the report directory.

### Details

Care should be used if `link.javascript` or `link.css` are set. If the javascript libraries for ReportingTools aren't linked, then it is likely that the resulting pages will not be as interactive as the default. The default css files are found in the `extdata/csslib` directory of the ReportingTools package, while the default javascript libraries are found in the `extdata/jslib` directory of the ReportingTools package. Both css and js files are copied from the package `extdata` directory into the `reportDirectory` of the HTMLReport.

### Value

An HTMLReport object with an open file handle to an HTML page.

### See Also

[HTMLReport-class](#), [publish](#)

### Examples

```
my.df <- data.frame(EGID = c("103", "104", "105", "106", "107"),
                    RPKM = c(4, 5, 3, 100, 75),
                    DE = c("Yes", "Yes", "No", "No", "No"))
html.report <- makeOldHTMLReport(shortName = "my_html_file",
                                reportDirectory = "reportDirectory")
publish(my.df, html.report)
```

---

<code>mockRnaSeqData</code>	<i>A counts table of mock RNA-seq data in mouse.</i>
-----------------------------	------------------------------------------------------

---

### Description

Expression data from an RNA-seq experiment. Rows were randomly assigned mouse Entrez ids.

### Usage

```
data(mockRnaSeqData)
```

### Examples

```
data(mockRnaSeqData)
```

---

 modifyReportDF-methods

*Decorate the Basic data.frame Representation of an Object*


---

### Description

This function is called after toReportDF (or the overriding.toDF if it is specified) in order to transform the basic data.frame into the form that will appear in the report. This typically includes transforming columns into links, adding image columns, etc. Each class publishable by ReportingTools should have either a custom objectToHTML method (which supercedes toReportDF and modifyReportDF) or a custom toReportDF and modifyReportDF methods which construct a dataframe to be placed in the report as an HTML table.

### Methods:

signature(df = "ANY", htmlRep = "ANY", object = "ANY") The default method is simply a placeholder, and returns the data.frame unchanged.

---

 objectToHTML-methods

*Transform R objects into HTML form to be inserted in HTML reports*


---

### Description

Unless overridden by a .toHTML argument, this function is called to transform objects being published into HTML form before they are inserted into a report.

### Methods

signature(object = "ANY") The default method for objectToHTML calls toReportDF (or the overriding .toDF) then modifyReportDF (or the overriding .modifyDF) to create a data.frame to be published. The data.frame method of objectToHTML is then called.

signature(object = "character") If the character vector contains HTML code (determined by attempted parsing), it is parsed but otherwise unchanged. If not, it is added as content to a new <p> HTML node.

signature(object = "data.frame") The data.frame is transformed into a filterable, sortable HTML table

signature(object = "ggbio") The graphic is written to the report's figure directory and an <image> tag to display the image is returned

signature(object = "ggplot") The graphic is written to the report's figure directory and an <image> tag to display the image is returned

signature(object = "recordedplot") The graphic is written to the report's figure directory and an <image> tag to display the image is returned

signature(object = "trellis") The graphic is written to the report's figure directory and an <image> tag to display the image is returned

---

publish-methods	<i>Methods for publishing a variety of data types in selected output formats</i>
-----------------	----------------------------------------------------------------------------------

---

## Description

These are a series of methods for taking various data types and coercing them into selected output formats.

## Methods

Data types can be output in several formats. Exactly what is done to coerce a data type into the given output format is described below.

**data.frame:**

`publish(object, htmlReport, tableTitle = NULL, ...)` The most basic object to publish to an HTMLReport is a data.frame. Most other methods involve coercing their objects to a data.frame and then calling `publish` on that data.frame. As such, this is where all styling for publishing tables can be centrally controlled. If `tableTitle` is specified, the title is printed above the table.

**MArrayLM:**

`publish(object, htmlReport, eSet, factor, n = 1000, pvalueCutoff = 0.01, lfc = 0, coef = NULL, adjust.method = 'BH', make.plots = TRUE, ...)` An MArrayLM object is coerced to a data.frame using a method similar to the `topTable` function from the limma package. The resulting table includes some selected feature data, and the log fold change and adjusted p-value from the linear model. The p-value adjustment method is set using the “adjust.method” argument. A glyph showing expression levels of each gene are also optionally plotted, based on the expression values from the ExpressionSet given in `eSet`, and the levels set in `factor`.

**DESeqDataSet:**

`publish(object, publicationType, factor = NULL, n = 1000, pvalueCutoff = 0.01, lfc = 0, contrast = NULL, resultName = NULL, make.plots = TRUE, ..., name)` To coerce a [DESeqDataSet](#) to a data.frame, we use the [results](#) function from the DESeq2 package, therefore [DESeq](#), or something similar must be run prior to `publish`. `contrast` and `resultName` are passed to the [results](#) function as the `contrast` and `name` parameters, please consult the documentation for [results](#) to see how these are specified. If present, `annotation.db` is used to find gene-level annotations for the rows in the [DESeqDataSet](#). Stripplots showing the expression values for each transcript are produced based on the normalized counts from the [DESeqDataSet](#), grouped by the levels in `factor`.

**DESeqResults:**

`publish(object, publicationType, DataSet = NULL, annotation.db = NULL, n = 500, pvalueCutoff = 0.01, lfc = 0, make.plots = TRUE, ..., name)` To coerce a [DESeqResults](#) object to a data.frame, we filter the results DataFrame such that the absolute log fold changes is greater than `lfc` and the adjusted p-value is less than `pvalueCutoff`. If present, `annotation.db` is used to find gene-level annotations for the rows in the [DESeqResults](#). If `make.plots` is TRUE, stripplots showing the expression values for each transcript are produced based on the normalized counts from the `DataSet`, which should be of class [DESeqDataSet](#), grouped by the levels in `factor`.

**DGEEExact:**

`publish(object, htmlReport, countTable, conditions, annotation.db = "org.Hs.eg", n = 1000, pvalueCutoff = 0.01, lfc = 0, adjust.method = "BH", make.plots = TRUE, ...)` DGEEExact objects are coerced to a data.frame using the `topTags` function from the edgeR

package and filtered base on `pvalueCutoff` and `lfc`. The resulting table includes feature data, derived from the annotation package defined in `annotation.db`, along with the fold change and adjusted p-value. For this to work, the feature names in the `DGEEExact` object and the `countTable` have to be the primary identifier from the annotation package. In most cases, the primary identifier will be Entrez Gene IDs. If no valid annotation package name is provided in `annotation.db`, then feature data will come from the “genes” slot in the `DGEEExact` object. The counts stored in `countTable` should be in units of counts per million provided by the `cpm` function in the `edgeR` package. The p-value adjustment method can be set using the “`adjust.method`” argument.

**DGELRT:**

```
publish(object, htmlReport, countTable, conditions, annotation.db = "org.Hs.eg",
n = 1000, pvalueCutoff = 0.01, lfc = 0, adjust.method = "BH", make.plots = TRUE, ...)
```

`DGELRT` objects are coerced to a `data.frame` using the `topTags` function from the `edgeR` package and filtered base on `pvalueCutoff` and `lfc`. The resulting table includes feature data, derived from the annotation package defined in `annotation.db`, along with the fold change and adjusted p-value. For this to work, the feature names in the `DGELRT` object and the `countTable` have to be the primary identifier from the annotation package. In most cases, the primary identifier will be Entrez Gene IDs. If no valid annotation package name is provided in `annotation.db`, then feature data will come from the “genes” slot in the `DGEEExact` object. The counts stored in `countTable` should be in units of counts per million provided by the `cpm` function in the `edgeR` package. The p-value adjustment method can be set using the “`adjust.method`” argument.

**HyperGResultBase, GOHyperGResult, PFAMHyperGResult:**

```
publish(object, htmlReport, selectedIDs, annotation.db, pvalueCutoff = 0.01, categorySize=10,
makePlot=FALSE, ...)
```

A `HyperGResult` object is coerced to a `data.frame` using a method similar to the `summary` function from the `Category` package. The resulting table includes for each classification the id, name, odds ratio of enrichment and p-value from the hypergeometric test. A glyph showing the level of overlap of each classification with the selected genes is also plotted. The number of genes found in this classification is listed in the table and links to another page with a table of the corresponding genes, symbols and names. An additional page listing the overlap genes is also linked to the main output. For a `GOHyperGResult` object, a plot depicting the relationship between the significant ontologies and their parents is also plotted if `makePlot` is `TRUE`. The `selectedIDs` are the Entrez ids of the genes of interest (i.e. from the gene universe); `annotation.db` is the species of the ids.

**GeneSetCollection:**

```
publish(object, htmlReport, annotation.db, setStats=NULL, setPValues=NULL, geneStats
= NULL, ..., .setToHTML = NULL, .setToDF = NULL, .modifySetDF = NULL)
```

A `GeneSetCollection` object is coerced to a `data.frame`. If `setStats` and/or `setPValues` are provided, they are included in the table of gene sets. The resulting table includes links to additional pages containing the ids, names and symbols of each gene in the corresponding set. To get the appropriate annotations for the individual gene sets, either `annotation.db` has to be the name of the appropriate annotation package, or the `geneIdType` of the individual gene sets has to have a non-empty annotation slot. The user can provide custom functions for coercing the enclosed `GeneSets` to their own preferred HTML representation by providing a function for `.setToHTML`. Likewise, a user-provided function can be passed in for `.setToDF`, to control the coercion of individual `GeneSets` to a `data.frame`. Additional modifications to the individual `GeneSet` `data.frames` can be provided as a list of functions in `.modifySetDF`.

**GeneSet:**

```
publish(object, htmlReport, annotation.db, geneStats = NULL, ...)
```

A `GeneSet` object is coerced to a `data.frame`. If `geneStats` are provided, an extra column of values for each of the `geneIds` in the `GeneSet` is appended to the table. The `data.frame` will optionally include feature annotations, such as Entrez IDs, gene symbols and gene names, if either `annotation.db` is a valid annotation package name or if the `annotation` slot of the `geneIdType` of the `GeneSet`

is a valid annotation package name.

trellis:

`publish(object, publicationType, figureTitle = NULL, filename = NULL, png.height = 480, png.width = 480, pdf.height = 7, pdf.width = 7, br = TRUE, ...)` Trellis objects, as created by lattice functions are published to an HTMLReport by first printing a pdf and png version of the object, and then including the png image in the HTML page, linking it to the pdf version of the plot.

Special cases:

There are a few special types that can be published to a HTMLReport. Publishing an HTMLReport to another HTMLReport will create a link from one report to the other, using the title of the published report as the link text.

**HTMLReport or HTMLReportRef to Data Package** There are two ways to publish R objects to a DataPackage: by using a character vector of object names, or by calling `publish` on the object directly, providing a name to use in saving the object. For most object types, basic documentation is created for the object in the data package, using the `promptData` function.

**CSVFile** data.frames can be published directly to csv files. Other data types, such as MArrayLM, DGEEExact, GOHyperGResult, PFAMHyperGResult can be output as CSVFile, by means of coercion to a data.frame first.

## Examples

```
my.df <- data.frame()
## html.report <- HTMLReport(shortName = "my_html_file",
##   reportDirectory = "reportDirectory")
# publish(my.df, html.report)
## finish(html.report)
```

---

readReport

*Read in an Existing Report*

---

## Description

This function is used to read an existing Report (previously created using ReportingTools) into R so that it can be modified or computed on.

## Usage

```
readReport(reportFile, handlers = fileHandlers(reportFile),
  .toHTML = NULL, .toDF = NULL, .modifyDF = NULL, title)
```

## Arguments

<code>reportFile</code>	A character value pointing to the HTML file containing an existing Report.
<code>handlers</code>	The ReportHandlers object(s) to attach to the resulting HTMLReportRef object.
<code>.toHTML</code>	The list of .toHTML overrides to attach to the HTMLReportRef object
<code>.toDF</code>	The list of .toDF overrides to attach to the HTMLReportRef object
<code>.modifyDF</code>	The list of .modifyDF overrides to attach to the HTMLReportRef object
<code>title</code>	Title to assign to the new HTMLReportRef object. If not specified this value is inferred from the file.

**Value**

An HTMLReportRef object representing the Report.

**Note**

Attempting to write the modified report to a new location without updating any report elements involving figures will result in broken image links because image locations are assumed to be relative to the location of the HTML file.

**Author(s)**

Gabriel Becker

**See Also**

[HTMLReportRef](#)

---

ReportHandlers-class    *Class "ReportHandlers"*

---

**Description**

A set of event-handler functions to be called when certain actions are performed on a HTMLReportRef object.

**Details**

ReportingTools ships with a number of predefined ReportHandlers constructorslo functions which generate ReportHandlers objects which implement commonly desired behaviors:

**fileHandlers** Accepts destination file location in the form of a character value, which is stored in the object's location slot. Report is built up internally and written to a file via the finish observer.

**fileWIndexHandlers** As with fileHandlers except that a table of contents is appended to the beginning of the report with links to each element within the page. Link text is the element names

**connectionHandlers** Accepts a connection to which the report should be streamed (which is stored in the location slot). Report is streamed out to the provided connection as elements are added to the report. codefinish completes the page and closes the connection.

**knitrHandlers** Accepts an option filename argument which will be stored as the location slot, but is not used internally. Elements are streamed in HTML form to standard output, which is captured by knitr and inserted into the page, allowing ReportingTools to be used as a formatting mechanism to publish objects within a Rmarkdown document and have them displayed correctly. Note: location MUST match the output file actually generated by the rmarkdown::render call (at least up to destination directory). If it does not the resulting page will not function properly.

**shinyHandlers** These handlers allow ReportingTools to be used as a formatting mechanism within a shiny Web application. Note in this case there is no meaningful HTML file being created that represents the report. location is set to NULL as no meaningful location exists in the shiny case.

## Objects from the Class

Objects can be created by calls of the form `new("ReportHandlers", ...)`.

## Slots

**init:** A function to be called when the report is initialized/created

**addElement:** A function to be called when an element is added/published to the report

**removeElement:** A function to be called when an element is removed from the report

**finish:** A function to be called when the report is "finished"

**args:** A named list containing zero or more of the elements "init", "addElement", "removeElement", and "finish", which contain additional arguments to be passed to the respective functions when called.

**location:** An R object indicating the location these handlers will publish the report to. The exact meaning of this argument depends on the behavior of the handlers.

## Methods

**path** `signature(object = "ReportHandlers")`: For file handlers, the file system location where the file will be written. For all others, NULL.

## Methods

**path** Returns the location associated with these ReportHandlers

## Author(s)

Gabriel Becker <gmbecker@ucdavis.edu>

## See Also

[HTMLReportRef](#)

## Examples

```
showClass("ReportHandlers")
```

---

reporting.theme

*Color theme for use in ReportingTools*

---

## Description

Getting some attractive colors for use in lattice graphics

## Usage

```
reporting.theme()
```

## Value

A list with slots as defined by the lattice function `standard.theme`

**See Also**

[standard.theme](#)

**Examples**

```
library(lattice)
theme <- reporting.theme()
lattice.options(default.theme = theme)
```

---

reporting.theme.alternate

*Alternate color theme for use in ReportingTools*

---

**Description**

Getting some alternate attractive colors for use in lattice graphics

**Usage**

```
reporting.theme.alternate()
```

**Value**

A list with slots as defined by the lattice function `standard.theme`

**See Also**

[standard.theme](#)

**Examples**

```
library(lattice)
theme <- reporting.theme.alternate()
lattice.options(default.theme = theme)
```

---

toReportDF-methods

*Class Specific Default Behavior for Transforming Objects to data.frames*

---

**Description**

The `toReportDF` generic and its methods are used to transform objects into a basic tabular (`data.frame`) form as part of the two step process of preparing to insert them into a report. This function (and functions passed to the `.toDF` argument used to override it) will only be called during the default ("ANY") `objectToHTML` method. Non-default `objectToHTML` methods are expected to perform the entire process of creating HTML code from an R object.

**Note**

The default `toReportDF` method is equivalent to calling `as.data.frame`.

**See Also**

[modifyReportDF](#)

---

validConnection	<i>Determine connection validity</i>
-----------------	--------------------------------------

---

**Description**

Determine if a connection to an HTML page can be written to or not.

**Usage**

```
validConnection(htmlRep)
```

**Arguments**

htmlRep            An object of class HTMLReport.

**Value**

Returns TRUE if the page can be written to. If the file handle has been closed, then FALSE.

**See Also**

[link{HTMLReport-class}](#)

**Examples**

```
my.df <- data.frame(EGID = c("103", "104", "105", "106", "107"),
                    RPKM = c(4, 5, 3, 100, 75),
                    DE = c("Yes", "Yes", "No", "No", "No"))
html.report <- makeOldHTMLReport(shortName = "my_html_file",
                                reportDirectory = "reportDirectory")
publish(my.df, html.report)
validConnection(html.report) # Returns TRUE

finish(html.report)
validConnection(html.report) # Returns FALSE
```

# Index

- \* **HTML**
  - Link, [10](#)
  - objectToHTML-methods, [13](#)
- \* **Link**
  - Link, [10](#)
- \* **Report**
  - objectToHTML-methods, [13](#)
- \* **classes**
  - BaseReport-class, [2](#)
  - CSVFile-class, [4](#)
  - DataPackage-class, [6](#)
  - HTMLReport-class, [8](#)
  - HTMLReportRef-class, [9](#)
  - ReportHandlers-class, [17](#)
- \* **data.frame**
  - toReportDF-methods, [19](#)
- \* **datasets**
  - mockRnaSeqData, [12](#)
- \* **methods**
  - filename-methods, [7](#)
  - finish-methods, [8](#)
  - modifyReportDF-methods, [13](#)
  - publish-methods, [14](#)
  - toReportDF-methods, [19](#)
- \* **publish**
  - objectToHTML-methods, [13](#)
- \* **report**
  - toReportDF-methods, [19](#)
- [[,HTMLReportRef,ANY,ANY-method  
(HTMLReportRef-class), [9](#)
- [[,HTMLReportRef,ANY-method  
(HTMLReportRef-class), [9](#)
- [[<-,HTMLReportRef,ANY,ANY,ANY-method  
(HTMLReportRef-class), [9](#)
- basePath (HTMLReport-class), [8](#)
- basePath,HTMLReport-method  
(HTMLReport-class), [8](#)
- BaseReport, [4](#), [7](#), [9](#)
- BaseReport-class, [2](#)
- baseUrl (HTMLReport-class), [8](#)
- baseUrl,HTMLReport-method  
(HTMLReport-class), [8](#)
- connectionHandlers  
(ReportHandlers-class), [17](#)
- CSVFile, [3](#), [3](#)
- CSVFile-class, [4](#)
- custHeaderPanel, [5](#)
- DataPackage, [3](#), [5](#)
- DataPackage-class, [6](#)
- dependencies (DataPackage-class), [6](#)
- dependencies,DataPackage-method  
(DataPackage-class), [6](#)
- dependencies<- (DataPackage-class), [6](#)
- dependencies<- ,DataPackage,character-method  
(DataPackage-class), [6](#)
- DESeq, [14](#)
- DESeqDataSet, [14](#)
- DESeqResults, [14](#)
- fileHandlers (ReportHandlers-class), [17](#)
- filename (filename-methods), [7](#)
- filename,CSVFile-method  
(filename-methods), [7](#)
- filename,HTMLReport-method  
(filename-methods), [7](#)
- filename-methods, [7](#)
- fileWIndexHandlers  
(ReportHandlers-class), [17](#)
- finish (finish-methods), [8](#)
- finish,DataPackage-method  
(finish-methods), [8](#)
- finish,HTMLReport-method  
(finish-methods), [8](#)
- finish,HTMLReportRef-method  
(finish-methods), [8](#)
- finish-methods, [8](#)
- HTMLReport, [3](#)
- HTMLReport (HTMLReportRef-class), [9](#)
- HTMLReport-class, [8](#)
- HTMLReportRef, [17](#), [18](#)
- HTMLReportRef-class, [9](#)
- knitrHandlers (ReportHandlers-class), [17](#)
- Link, [10](#)

- Link, character-method (Link), 10
- Link, HTMLReportRef-method (Link), 10
- Link, list-method (Link), 10
- Link-methods (Link), 10
  
- makeOldHTMLReport, 11
- mockRnaSeqData, 12
- modifyReportDF, 20
- modifyReportDF
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, ANY-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, DESeqDataSet-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, DESeqResults-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, DGEExact-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, DGELRT-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, GeneSet-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, GOHyperGResult-method
  - (modifyReportDF-methods), 13
- modifyReportDF, ANY, ANY, MArrayLM-method
  - (modifyReportDF-methods), 13
- modifyReportDF-methods, 13
  
- name (BaseReport-class), 2
- name, BaseReport-method
  - (BaseReport-class), 2
- name<-, BaseReport, character-method
  - (BaseReport-class), 2
- name<-, CSVFile, character-method
  - (CSVFile-class), 4
- name<-, HTMLReport, character-method
  - (HTMLReport-class), 8
  
- objectToHTML (objectToHTML-methods), 13
- objectToHTML, ANY-method
  - (objectToHTML-methods), 13
- objectToHTML, character-method
  - (objectToHTML-methods), 13
- objectToHTML, data.frame-method
  - (objectToHTML-methods), 13
- objectToHTML, ggbio-method
  - (objectToHTML-methods), 13
- objectToHTML, ggplot-method
  - (objectToHTML-methods), 13
- objectToHTML, recordedplot-method
  - (objectToHTML-methods), 13
- objectToHTML, trellis-method
  - (objectToHTML-methods), 13
  
- objectToHTML, XMLInternalNode-method
  - (objectToHTML-methods), 13
- objectToHTML-methods, 13
  
- page (HTMLReport-class), 8
- page, HTMLReport-method
  - (HTMLReport-class), 8
- page<- (HTMLReport-class), 8
- page<-, HTMLReport, character-method
  - (HTMLReport-class), 8
- pageWithSidebar, 5
- path (BaseReport-class), 2
- path, BaseReport-method
  - (BaseReport-class), 2
- path, DataPackage-method
  - (DataPackage-class), 6
- path, HTMLReport-method
  - (HTMLReport-class), 8
- path, HTMLReportRef-method
  - (HTMLReportRef-class), 9
- path, ReportHandlers-method
  - (ReportHandlers-class), 17
- publish, 12
- publish (publish-methods), 14
- publish, ANY, DataPackage-method
  - (publish-methods), 14
- publish, ANY, HTMLReport-method
  - (publish-methods), 14
- publish, ANY, HTMLReportRef-method
  - (HTMLReportRef-class), 9
- publish, ANY, list-method
  - (publish-methods), 14
- publish, character, DataPackage-method
  - (publish-methods), 14
- publish, data.frame, CSVFile-method
  - (publish-methods), 14
- publish, data.frame, DataPackage-method
  - (publish-methods), 14
- publish, data.frame, HTMLReport-method
  - (publish-methods), 14
- publish, data.frame, HTMLReportRef-method
  - (publish-methods), 14
- publish, DESeqDataSet, HTMLReportRef-method
  - (publish-methods), 14
- publish, DESeqResults, HTMLReportRef-method
  - (publish-methods), 14
- publish, DGEExact, ANY-method
  - (publish-methods), 14
- publish, DGEExact, HTMLReport-method
  - (publish-methods), 14
- publish, DGEExact, HTMLReportRef-method
  - (publish-methods), 14

- publish,DGELRT,ANY-method  
(publish-methods), 14
- publish,DGELRT,HTMLReport-method  
(publish-methods), 14
- publish,DGELRT,HTMLReportRef-method  
(publish-methods), 14
- publish,GeneSet,HTMLReport-method  
(publish-methods), 14
- publish,GeneSetCollection,ANY-method  
(publish-methods), 14
- publish,GeneSetCollection,HTMLReport-method  
(publish-methods), 14
- publish,GeneSetCollection,HTMLReportRef-method  
(publish-methods), 14
- publish,GOHyperGResult,ANY-method  
(publish-methods), 14
- publish,GOHyperGResult,HTMLReport-method  
(publish-methods), 14
- publish,GOHyperGResult,HTMLReportRef-method  
(publish-methods), 14
- publish,HTMLReport,HTMLReport-method  
(publish-methods), 14
- publish,HTMLReport,HTMLReportRef-method  
(publish-methods), 14
- publish,HyperGResultBase,HTMLReport-method  
(publish-methods), 14
- publish,HyperGResultBase,HTMLReportRef-method  
(publish-methods), 14
- publish,list,HTMLReport-method  
(publish-methods), 14
- publish,list,HTMLReportRef-method  
(publish-methods), 14
- publish,MArrayLM,ANY-method  
(publish-methods), 14
- publish,MArrayLM,HTMLReport-method  
(publish-methods), 14
- publish,MArrayLM,HTMLReportRef-method  
(publish-methods), 14
- publish,PFAMHyperGResult,ANY-method  
(publish-methods), 14
- publish,PFAMHyperGResult,HTMLReport-method  
(publish-methods), 14
- publish,PFAMHyperGResult,HTMLReportRef-method  
(publish-methods), 14
- publish,trellis,HTMLReport-method  
(publish-methods), 14
- publish,trellis,HTMLReportRef-method  
(publish-methods), 14
- publish-methods, 14
- readReport, 16
- reportDirectory (BaseReport-class), 2
- reportDirectory<- (BaseReport-class), 2
- reportDirectory<- ,BaseReport,character-method  
(BaseReport-class), 2
- ReportHandlers, 9, 10
- ReportHandlers-class, 17
- reporting.theme, 18
- reporting.theme.alternate, 19
- results, 14
- shinyHandlers (ReportHandlers-class), 17
- show,BaseReport-method  
(BaseReport-class), 2
- show,HTMLReport-method  
(HTMLReport-class), 8
- show,HTMLReportRef-method  
(HTMLReportRef-class), 9
- standard.theme, 19
- title (BaseReport-class), 2
- title,BaseReport-method  
(BaseReport-class), 2
- title<- (BaseReport-class), 2
- title<- ,BaseReport,character-method  
(BaseReport-class), 2
- toReportDF (toReportDF-methods), 19
- toReportDF,ANY-method  
(toReportDF-methods), 19
- toReportDF,data.frame-method  
(toReportDF-methods), 19
- toReportDF,DESeqDataSet-method  
(toReportDF-methods), 19
- toReportDF,DESeqResults-method  
(toReportDF-methods), 19
- toReportDF,DGEEExact-method  
(toReportDF-methods), 19
- toReportDF,DGELRT-method  
(toReportDF-methods), 19
- toReportDF,GeneSet-method  
(toReportDF-methods), 19
- toReportDF,GeneSetCollection-method  
(toReportDF-methods), 19
- toReportDF,GOHyperGResult-method  
(toReportDF-methods), 19
- toReportDF,MArrayLM-method  
(toReportDF-methods), 19
- toReportDF,PFAMHyperGResult-method  
(toReportDF-methods), 19
- toReportDF-methods, 19
- url,HTMLReport-method  
(HTMLReport-class), 8

validConnection, [20](#)