# Basics of ReportingTools

# Jason A. Hackney and Jessica L. Larson November 4, 2025

# Contents

1	Introduction	2
2	Basics of Reporting	2
3	Adding plots or text to a report	3
4	Adding plots or links to a report table	5
5	Multiple Tables to the same page	6
6	Publishing other types of data and more advanced features	7
7	References	7

#### 1 Introduction

Frequently, when performing an analysis, it is helpful to be able to share these results in several formats at once: as HTML tables, csv files or even as R data packages. ReportingTools attempts to make this as painless as possible. At its heart, ReportingTools is based on a number of pieces of interlocking machinery that transform popular Bioconductor objects into reports.

In this vignette we will highlight the fundamentals of ReportingTools. ReportingTools has several methods for displaying microarray and RNA-seq results and can also be incorporated into shiny applications and knitr reports; for more details, please refer to the corresponding vignettes (knitr.Rmd and shiny.Rnw, respectively) or the ReportingTools site.

For more information on ReportingTools, please see Huntley, Larson, et al. (2013).

### 2 Basics of Reporting

The easiest type of report to generate is a csv file. This is done using the CSVFile class and the publish method. To start we'll create a data.frame that we'll use throughout the vignette.

```
> 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"))

> my.df

EGID RPKM DE

1 103 4 Yes

2 104 5 Yes

3 105 3 No

4 106 100 No

5 107 75 No
```

Next, we'll create the CSVFile object to which we'll publish our results. We output the results to a new directory called reports. Note that ReportingTools will create this directory for you if it does not exist already.

```
> library(ReportingTools)
> csvFile <- CSVFile(shortName = "my_csv_file",
+ reportDirectory = "./reports")
> publish(my.df, csvFile)
```

Obviously, this isn't much less work than just calling write.csv on the data.frame itself, but this is really just a toy example. We can also publish the data.frame as an HTML report.

```
> htmlRep <- HTMLReport(shortName = "my_html_file",
+ reportDirectory = "./reports")
> publish(my.df, htmlRep)
> finish(htmlRep)
```

It's necessary to call finish on the HTMLReport, to allow the contents to be written to the file.

It's also possible to publish the same object in two separate formats at once.

```
> csvFile2 <- CSVFile(shortName = "my_csv_file2",
+ reportDirectory = "./reports")
> htmlRep2 <- HTMLReport(shortName = 'my_html_file2',</pre>
```

#### my\_html\_file

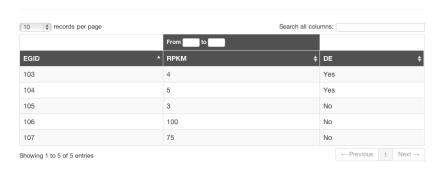


Figure 1: Resulting page created by publish for my.df.

```
title="Publishing a data frame and csv file together",
      reportDirectory = "./reports")
> publish(my.df, list(csvFile2, htmlRep2))
> finish(htmlRep2)
```

The same few lines of code could be used to publish, for example, the results of a limma differential expression analysis, or the results of a Gene Ontology analysis, all without worrying about coercing the objects to a tabular format ourselves. For more information, see the microarray and RNA-seq vignettes.

#### 3 Adding plots or text to a report

To add links, additional text or plots to a report, simply open the report with HTMLReport, write to it via the publish function and then call publish on the original data frame and finish the report. Below we make a simple plot and then add it and some descriptive text to our report.

```
> plot(my.df$EGID, my.df$RPKM, xlab="EGID",
         ylab="RPKM", main="Scatter plot of RPKMs", col="blue")
> scatterPlot <- recordPlot()
> library(lattice)
> barPlot <- barchart(my.df$RPKM~my.df$EGID) ##lattice plots behave slightly differently
> htmlRep3 <- HTMLReport(shortName = "my_html_file3", title="Adding a plot directly to the p
      reportDirectory = "./reports")
> publish(scatterPlot, htmlRep3, name = "scatterPlot")
> publish("This is a bar plot", htmlRep3)
> publish(barPlot, htmlRep3, name = "barPlot")
> publish(my.df, htmlRep3, name="Table")
> finish(htmlRep3)
```

We can also publish existing images and text directly to sites with hwriter.

```
> png(filename="reports/barplot.png")
> barplot(my.df$RPKM, names.arg=my.df$EGID, xlab="EGID",
          ylab="RPKM", main="Bar plot of RPKMs", col="blue")
+
> dev.off()
> library(hwriter)
```

# This is a bar plot Search all columns: 10 \$ records per page From RPKM **EGID** 103 DE Yes 4 5 3 104 Yes 105 No 106 100 No 107 75 No ← Previous | 1 | Next → Showing 1 to 5 of 5 entries

Adding a plot directly to the page

Figure 2: Resulting page created after adding additional figures and text with publish.

### 4 Adding plots or links to a report table

To add additional plots or links to a report table, we can create a new data frame with the path to the plots and our links of interest. We then publish this data frame.

Below we make a set of simple plots and then add the images along with new links to the NCBI gene database to our data frame.

We can also update our data frame by editing, adding and removing columns with functions. We then include these functions in our publish call as a list with .modifyDF and .toHTML..modifyDF uses the basic data frame as its default object and then modifies it with the corresponding function.

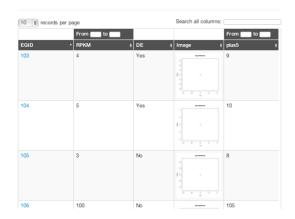


Figure 3: Resulting page created after adding figures and links to table with .modifyDF.

```
object$Image <- hwriteImage(imagename, link = imagename, table = FALSE, height=150
          return(object)
> ##This function removes the link column
> removeLink <- function(object, ...) {</pre>
          object <- subset(object, select = -Link)
+
          return(object)
> ##This function links the EGID column to the entrez database
> addEGIDLink <- function(object, ...) {</pre>
          object$EGID <- hwrite(as.character(object$EGID),</pre>
                                 link = paste0("http://www.ncbi.nlm.nih.gov/gene/",
+
                                   as.character(object$EGID)), table = FALSE)
          return(object)
 htmlRep6 <- HTMLReport(shortName = "my_html_file6",</pre>
                          title = "Manipulating the data frame directly",
                          reportDirectory = "./reports")
> publish (my.df, htmlRep6,
          .modifyDF = list(add5, makeNewImages, removeLink, addEGIDLink))
> finish(htmlRep6)
```

## 5 Multiple Tables to the same page

It is also possible to publish multiple tables to the same html page. We can change the order of the tables via pos.

```
> df2 \leftarrow data.frame(x = 1:5, y = 11:15)
> df3 \leftarrow data.frame(x = c("a", "b", "c"), y = 1:3)
> htmlRep7 \leftarrow HTMLReport(shortName = "my_html_file7", title = "Many tables, one page", eportDirectory = "./reports")
> publish(my.df, htmlRep7,
```

#### Publishing objects that are not data frames



Figure 4: Resulting page created after transforming a matrix to a data frame with .toDF.

```
+ .modifyDF = list(add5, makeNewImages, removeLink, addEGIDLink),
+ name = "Df1")
> publish(df2, htmlRep7, name = "Df2")
> publish(df3, htmlRep7, name = "Df3", pos = 2)
> finish(htmlRep7)
```

### 6 Publishing other types of data and more advanced features

To publish data that is not a data frame, there is a need to create and use a .toDF function. For example, suppose we have a matrix we would like to publish. ReportingTools will convert the basic matrix to a data.frame and then publish it.

For publishing experimental results, including how to publish a limma-based linear model and a edgeR objects, please see the relevant vignettes. There are built-in ReportingTools methods to publish non-data frame objects typically encountered in microarray and RNA-seq analyses. Example output is shown below.

### 7 References

Huntley, M.A., Larson, J.L., Chaivorapol, C., Becker, G., Lawrence, M., Hackney, J.A., and J.S. Kaminker. (2013). ReportingTools: an automated results processing and presentation toolkit for high throughput genomic analyses. *Bioinformatics*. **29**(24): 3220-3221.

### Analysis of BCR/ABL translocation differential expression

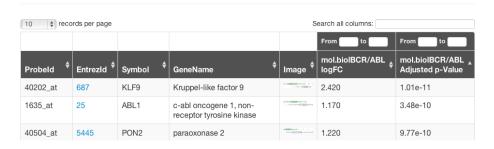


Figure 5: Resulting page created for analysis of a microarray study with limma.