The 'graphicsQC' Package

November 9, 2008

Type Package

Title Quality Control for Graphics in R

Version 0.9

Date 2008-07-24

Author Stephen Gardiner

Maintainer Stephen Gardiner <sgar060@aucklanduni.ac.nz>

Depends XML

Suggests Sxslt

Description The package provides functions to generate graphics files, compare them with "model" files, and report the results.

License GPL-2

URL http://graphicsqc.r-forge.r-project.org

R topics documented:

`graphicsQC-package` *Quality Control for Graphics*

Description

Generates graphics files, compares them with "model" files, and reports the results.

Details

Package:	graphicsQC
Type:	Package
Version:	0.9
Date:	2008-10-30
License:	GPL-2

To generate files, use `plotExpr`, `plotFile`, or `plotFunction`. To compare sets of these, use `compare`. To generate a report based on the comparison, use `writeReport`.

One possible way of using these functions is to create a set of plots in a directory in an old version of R (say, the control group) using one of the plotting functions. Then to load a new version of R and create the same plots in a different directory (say, the test group). A comparison can then be done by specifying the control and test directories. Then a report can be made on the comparison object.

It is highly recommended to use separate directories for the test and control. If the same directory is used for both, all the prefixes in the test and all the prefixes in the control must be unique, and auto-detect will not work if the same directory is given twice.

Author(s)

Stephen Gardiner

References

Free Software Foundation, Inc. 2008 *Diffutils*. `http://www.gnu.org/software/diffutils/diffutils.html`

ImageMagick Studio LLC. 2008 *ImageMagick*. `http://www.imagemagick.org/`

Murrell, P. & Hornik, K. 2003 *Quality Assurance for Graphics in* R `http://www.ci.tuwien.ac.at/Conferences/DSC-2003/Proceedings/MurrellHornik.pdf`.

See Also

`plotExpr`, `plotFile`, `plotFunction`, `compare`, `writeReport`

Examples

```
## Not run:
  # Create some plots to compare (1st and 3rd plots have differences)
  comp1 <- plotExpr(c("plot(1:10)", "plot(4:40)", "x<-3", "plot(2:23)"),
                    c("pdf", "ps"), "myPrefix", "comp1")
  comp2 <- plotExpr(c("plot(1:11)", "plot(4:40)", "x<-3", "plot(5:15)"),
                    c("pdf", "ps"), "myPrefix", "comp2")

  # Compare them
  compExpr <- compare(comp1, comp2)

  # Write a HTML report
  writeReport(compExpr)
## End(Not run)
```

`plotExpr` *Plot arbitrary code*

Description

Produce plots from R expression(s), function(s), or file(s) in specified file formats. An XML file is also created which contains information about the plots produced.

Usage

```
plotExpr(expr, filetype = NULL, path = NULL, prefix = NULL,
         clear = FALSE)

plotFile(filename, filetype = NULL, path = NULL, prefix = basename(filename),
         clear = FALSE)

plotFunction(fun, filetype = NULL, path = NULL, prefix = fun,
             clear = FALSE)
```

Arguments

`expr`	character vector of R expressions which may or may not produce graphical output.
`filename`	the name of the file which the expressions are to be read from. The path is assumed to be relative to the current working directory unless an absolute path is given.
`fun`	character vector naming the function(s) to plot or just the (named) function.
`filetype`	character vector specifying file formats to produce the plots in (see details for currently supported formats).
`path`	character vector; path to produce output in. If not given, the current working directory will be used.
`prefix`	character vector; prefix for files produced. If multiple functions or files are given, the resulting plotFile or plotFunction XML file will use the first prefix.
`clear`	logical (not NA); remove files with names we might use first. If `clear` is `FALSE` and files exist with names we might use, an error is given.

Details

All functions evaluate the code they are given, capturing and recording any warnings and errors. The code run for plotFunction is extracted from any example code (see `example`).

If an error is encountered when running a block of code, that particular block (say, a file or function) will stop being executed for that filetype but the error will be recorded. If a warning is encountered, the code will continue being evaluated.

The name for the log file is based on the first prefix and the type of function producing the plots. For `plotExpr` XML logs, 1 log file will be produced with a name like 'prefix-log.xml'. `plotFile` and `plotFunction` work by making multiple calls to `plotExpr`, so will produce `plotExpr` logs (one for each file or function respectively), as well as their own log,

which will be named (using the first prefix) like 'prefix-fileLog.xml' and 'prefix-funLog.xml' respectively.

Currently supported file formats are 'pdf', 'png', and 'ps'.

Value

A list of class `qcPlotExprResult`, `qcPlotFileResult`, or `qcPlotFunctionResult` respectively. The list contains information about the environment creating the plots (Operating System, R version, date, call), the names of the plots produced and any warnings/errors produced.

Warning

Do not give any code that will open a graphics device (especially if that device is not closed).

See Also

`compare`, `writeReport`.

Examples

```
## Not run:
  # plotExpr example:
  example1 <- plotExpr(c("plot(1:10)", "plot(4:40)", "x<-3", "plot(2:23)"),
                       c("pdf", "ps"), "example1", "myPrefix")
  # There should now be a folder "example1" in the current
  # working directory containing pdf and ps files and myPrefix-log.xml, ie
  list.files("example1")

  # plotFunction example:
  example2 <- plotFunction(c("plot", "barplot", "hist"), c("pdf", "ps"),
                           path = "example2")
  list.files("example2")

  # A bigger example:
  # require(grid)
  # gridExample <- plotFunction(ls("package:grid"), c("pdf", "png", "ps"),
  #                             path = "gridExample")
## End(Not run)
```

`compare` *Compare graphics output*

Description

Compares plots/warnings/errors from `plotExpr`, `plotFile`, or `plotFunction`. For the text-based formats (i.e. pdf or ps), a .diff file is created. If ImageMagick is installed, plots of the differences will also be produced.

Usage

```
compare(test, control, path = test$info$directory,
        erase = c("none", "identical", "files", "all"))
```

Arguments

`test, control` either:

- R objects of class `qcPlotExpression`, `qcPlotFile`, or `qcPlotFunction`.
- Character vectors of the paths to the respective files, where relative paths are assumed unless an absolute path is given.
- Character vectors of the directories which contain the log files to compare. The highest classed object in the folder will be chosen for comparison (i.e. if a plotFunction log is in the directory and many plotExpr logs, all of the plotExpr logs will be assumed to belong to the plotFunction log).

The specification for `test` and `control` can be mixed and matched, as long as the resulting objects are of the same class.

`path` character vector; specifies where all the diff output (.diff files, plots of differences, and comparison logs) should be placed.

`erase` character vector; one of `"none"`, `"identical"`, `"files"`, or `"all"`.

`"none"` do not delete anything.

`"identical"` delete plots in the `test` directory which were identical.

`"files"` delete all plots (and .diff files) in the `test` directory (leaving only the log files).

`"all"` delete all files created in the `test` directory and then the directory if it is empty

Currently only `"none"` is fully supported.

Details

Plots are compared using GNU diff. If a difference is detected and the current filetype being compared is a text-based format, a .diff file will be produced. If ImageMagick is installed, plots of differences will also be created.

It is possible for some plots to appear say, in the test group, but not in the control group (i.e. the function `plot` has an extra example plot in a new version of R). These such plots are classified as 'unpaired'. Unpaired files do not have a corresponding plot to compare with so are separated into an unpaired section. It is also possible for entire filetypes to be unpaired. Currently if there is a completely unpaired function or file when trying to compare, recycling will be used. This is intended to change in the future.

In many instances, it is also useful to know whether there is any change in warnings or errors. If any difference is detected in the warnings/errors for a filetype, all of the warnings or errors (whichever had the difference detected) for that filetype are given. It is then up to the user to decide what the difference is (i.e. whether the ordering has changed or if one group has an extra warning etc.).

For each set of plot-logs being compared, a comparison log will be produced. So for each pair of qcPlotExprResult logs being compared, a comparison log will be produced with a name like 'testPrefix+controlPrefix-compareExprLog.xml'. When comparing qcPlotFileResults or qcPlotFunctionResults there will also be a compareFileLog or compareFunLog produced which will take a name like 'testPrefix+controlPrefix-compareFunLog.xml', where the testPrefix and controlPrefix are chosen from the first prefixes in the set of compareExprLogs being compared (which in turn come from the first plotExpr logs). These logs are placed in `path`.

Value

A list of class `qcCompareExprResult`, `qcCompareFileResult` or `qcCompareFunResult` containing the results of the comparisons.

`qcCompareExprResult` files contain a list of info about the Operating System, R version, date, call, the info from the test, info from the control, and then information about the results of the comparisons (results by filetype giving the result, names of diff files and plots of differences if produced), including any unpaired plots or filetypes (with corresponding warnings/errors).

For `qcCompareFile` or `qcCompareFun` an initial info section is included, followed by a list containing each individual `qcCompareExprResult`.

Note

GNU diff must be installed on the system. ImageMagick is not necessary, but greatly extends functionality.

See Also

`plotExpr`, `plotFile`, `plotFunction`, `writeReport`

Examples

```
## Not run:
  # Create sets to compare (1st and 3rd are different)
  comp1 <- plotExpr(c("plot(1:10)", "plot(4:40)", "x<-3", "plot(2:23)"),
                    c("pdf", "ps"), "myPrefix", "comp1")
  comp2 <- plotExpr(c("plot(1:11)", "plot(4:40)", "x<-3", "plot(5:15)"),
                    c("pdf", "ps"), "myPrefix", "comp2")
  compExpr <- compare(comp1, comp2)
  # All the diff output has been placed in "comp1" (the test directory)
  compExpr
  # For a better way of viewing this, see ?writeReport
## End(Not run)
```

`writeReport` *Generate a HTML report based on plots or comparisons*

Description

Will produce a HTML report of the results from any of the `qcPlot*`, or `qcCompare*` results.

Usage

```
writeReport(qcResult, xslStyleSheets = NULL)
```

Arguments

`qcResult` one of:

- an R object of class `qcPlot*`, or `qcCompare*`.
- the path to the log file to report on

- a path to the directory, where the highest classed log file will be auto-detected and then reported on (note that first comparison logs are searched for, then plot logs).

`xslStyleSheets`
: a named list specifying which XSL style sheets to override by giving the name of the style sheet to override, and the location of the xsl file. Can override any of: "plotExprStyleSheet", "plotFunAndFileStyleSheet", "compareExprStyleSheet", and "compareFunAndFileStyleSheet". If none are specified, the default (system) ones are used.

Details

When reporting on an object, all further qcPlot* or qcCompare* files which the current object refers to are also reported on. This is so that full information reports can be given, along with individual breakdowns. In order for this to happen, all log files that the object currently being reported on refers to must exist, as well as any subsequent log files that they refer to.

All reports are placed in the same directory as the XML file they refer to, with the same name, except with the extension changed from '.xml' to '.html'.

Value

A character vector giving the (absolute) path of the highest classed object reported on. Comparison logs are considered higher classed than plot logs.

See Also

`plotExpr`, `plotFile`, `plotFunction`, `compare`.

Examples

```
## Not run:
  # After running the `?compare' example
  writeReport(compExpr)

  # Showing how to overwrite stylesheets
  # writeReport(compExpr, list(compareExprStyleSheet="~/myCompareExpr.xsl"))
## End(Not run)
```