Quality Control for Graphics in R

The **graphicsQC** package

Stephen Gardiner

BSc(Hons) Project
Supervised by Dr. Paul Murrell

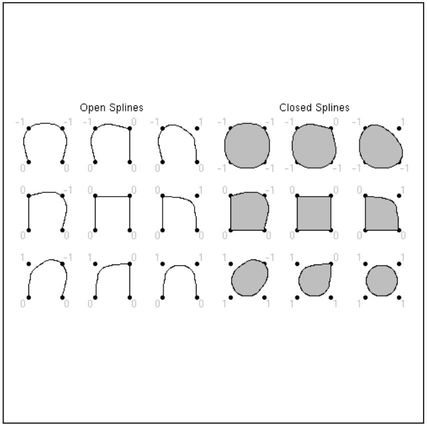
Department of Statistics
The University of Auckland

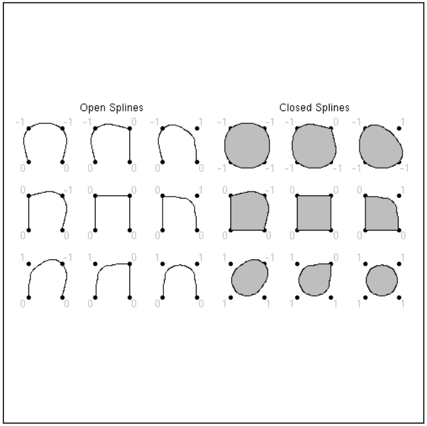
1. Motivation
2. Quality Control
3. The graphicsQC package
    - Plotting arbitrary expressions
    - Comparing sets of plots
    - Reporting on whats been done
4. Conclusions

Motivation

- New version of R or some plotting functions improved
    - Are the plots created still the same??

Is there a difference?





We'll come back to this.

What is “Quality Control”?

- Really just software testing
  - No errors
  - Correct output
- For graphics, this is ensuring the output stays the same as some initial, correct (control) output
  - This is called *regression testing*

Existing QC for R

Surely R already has some method of checking the graphics?

- `R CMD check` only checks code and output from the code — no graphics checking
- `make check` has some hard-coded examples for PostScript only
- A similarly named **graphicsQC** package Paul made. . .

Paul's **graphicsQC** package

- Really just a proof-of-concept package
- Ran `example()` code from functions into a directory then compared
- Listed files which were different — no other information reported except for `png` format which had plots of *just* the differences
- Very little error checking
- Needed a complete re-write. . .

A motivating example

Between revisions 44416 and 44417 of R (when version 2.7.0 was under development), an anisotropy change was made to R. A change was expected for the raster formats (i.e. `png`), but not for the vector formats (i.e. `PostScript and PDF`). It was of interest to see what changes this may have on the **grid** package.

The **graphicsQC** package

Three main things need to be done:

- Create plots (in possibly many file formats) — so evaluate *any* code and record which plots were created
- Compare sets of plots
- Report the results

Each of these will now be discussed in turn.

Plotting arbitrary expressions

Created function `plotExpr()`. Evaluate code for each graphics device specified

- continue on warnings
- stop evaluation for given device on errors, but continue functioning $\rightarrow$ uses `tryCatch` mechanism

i.e.
```
firstExample <-
 plotExpr(c("y=10", "x=5", "plot(1:10)"),
             filetype = c("pdf", "ps", "png"),
             prefix = "myPlots",
             path = "/tmp/myDir")
```
Records information, warnings, errors, and which plots were created into an XML ‘log’ file.

```
> firstExample

plotExpr Result:
Call:  plotExpr(c("y=10", "x=5", "plot(1:10)"),
filetype = c("pdf", "ps", "png"), prefix =
"myPlots", path = "/tmp/myDir")
R version:  R version 2.6.2 (2008-02-08)
Directory:  /tmp/myDir
Filename:  myPlots-log.xml
Formats:
  pdf :  Plots: myPlots-1.pdf
  ps :   Plots: myPlots-1.ps
  png :  Plots: myPlots-1.png
```

Example XML log produced

```
<?xml version="1.0"?>
<qcPlotExprResult>
 <info>
  <OS>unix</OS>
  <Rver>R version 2.6.2 (2008-02-08)</Rver>
  <date>Wed Nov 19 15:40:48 2008</date>
  <call>
   <![CDATA[
   plotExpr(c("y=10", "x=5", "plot(1:10)"),
filetype = c("pdf",     "ps", "png"), prefix =
"myPlots", path = "/tmp/myDir")   ]]>
  </call>
  <directory>/tmp/myDir</directory>
  <logFilename>myPlots-log.xml</logFilename>
 </info>
```

```
<plots type="pdf">
  <warnings/>
  <error/>
  <plot>myPlots-1.pdf</plot>
 </plots>
<plots type="ps">
  <warnings/>
  <error/>
  <plot>myPlots-1.ps</plot>
 </plots>
 <plots type="png">
  <warnings/>
  <error/>
  <plot>myPlots-1.png</plot>
 </plots>
</qcPlotExprResult>
```

We want to plot more than just a couple of expressions. Plotting files and functions are just extensions of expressions.

- `plotFile()` - one file is one set of "expressions", 2 files are 2 sets, etc.
- `plotFunction()` - one call to `example()` for each function. Each call is an expression.

So each just makes multiple calls to `plotExpr()`! Both produce similar XML log files with links (paths) to each set of expressions produced.

grid example

So, to plot the functions from **grid**:

```
> grid44417 <-
   plotFunction(ls("package:grid"),
                filetype = c("pdf", "ps", "png"),
                path="~/tests/R44417")
```

Comparing sets of plots

- GNU `diff` is used to compare plots
- Plots are pairwise compared by filetype with ‘unpaired’ files split into a separate section
- Any warnings and errors compared
- Plots highlighting the differences created if files are different and ImageMagick is installed
- Results stored in XML log files

Comparing the **grid** functions:

```
> gridCompare <- compare(test = grid44417,
                         control = "~/tests/R44416/")
```

Reporting

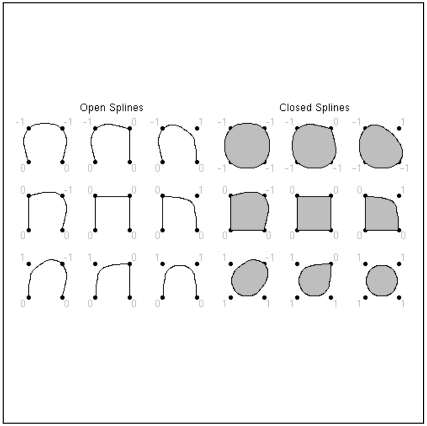
- We need a dynamically generated report to present the information from plots and comparisons.
- Data stored in XML with *links* to other files
- HTML the obvious choice?
    - Aiding this, XSL language is designed specifically for transforming XML into other formats (typically HTML)
    - **Sxslt** package to use XSL from R

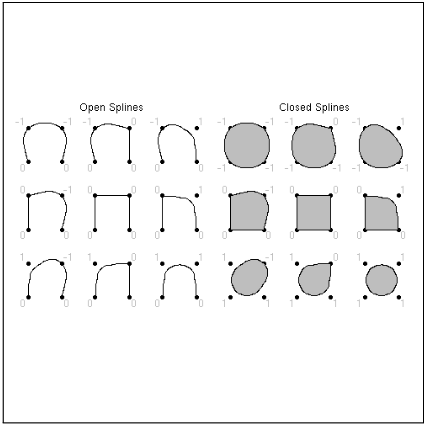
Reporting on grid

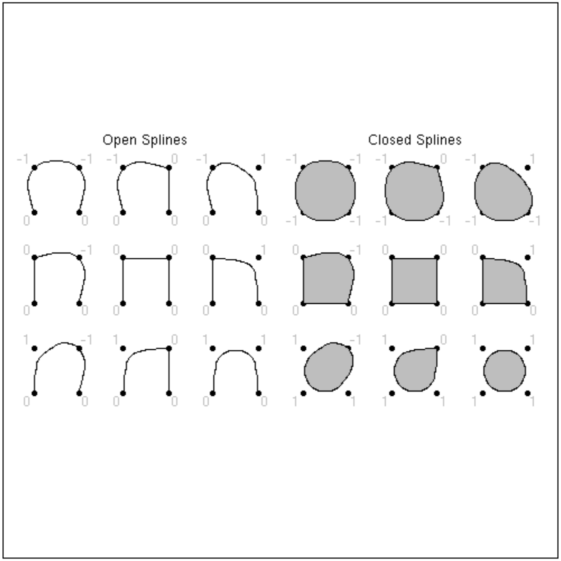
```
> writeReport(gridCompare)
[1] "/home/stephen/tests/R44417/absolute.size-compareFunLog.html
```

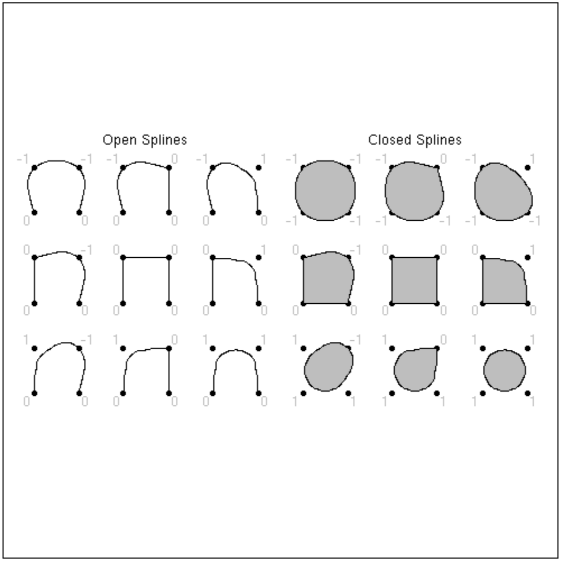
Here's one I prepared earlier. . .

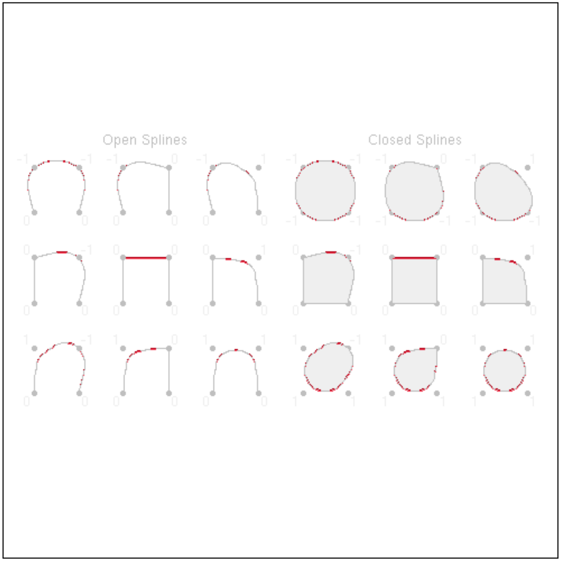
The answers to the first example






Open Splines
Closed Splines
-1
0
1


Open Splines
Closed Splines


Open Splines
Closed Splines

Conclusions

- Quality Control for Graphics in R can now be a more automated process.
- Open standards used at all stages — users can take the process out of R wherever they want, whenever they want.
- Useful for detecting changes but cannot identify *why* the changes occur.
- Future improvements include `plotPackage()` and platform independence.

Acknowledgements

- I would like to thank Paul for his ongoing guidance, support, and positive enthusiasm.
- Duncan Temple Lang for the **XML** and **Sxslt** packages on which **graphicsQC** depends and suggests respectively
- ImageMagick Studio

Any Questions?