Refactoring the xtable Package

David J Scott\textsuperscript{1}  \quad Daniel Geals\textsuperscript{1}  \quad Paul Murrell\textsuperscript{1}

\textsuperscript{1}Department of Statistics, The University of Auckland

July 10, 2015
Outline

1. Introduction

2. Analysis

3. Testing
Outline

1. Introduction
2. Analysis
3. Testing
Outline

1 Introduction

2 Analysis

3 Testing
Introduction

- `xtable` was written by David Dahl
- A number of others have contributed code
- I am now the maintainer
- `xtable` outputs formatted text to produce tables for inclusion in \LaTeX{} and HTML documents (hence also markdown)
- The production of HTML is far less developed
- `xtable` is widely used:
  - in the top 50 downloaded packages on CRAN
  - 30 packages depend on `xtable`
  - 70 packages either import or suggest `xtable`
- I will mainly concentrate on the \LaTeX{} side of `xtable` in this talk, although I am actually very interested in developing the HTML capability of `xtable`
Introduction

- The \LaTeX{} side produces tables of two sorts
  - function xtable has methods for objects of various classes: matrix, data.frame, lm, anova, aov, ts, ...
  - via user-specified table formatting with arguments supplied to the print.xtable() function

- The former problem is more specific and only requires a function to be written for a given class

- The second problem is less well defined
  - there are a number of extensions of the basic tabular environment: tabularx, tabulary, tabu, array
  - there is the package booktabs which changes some vertical spacing and allows for differently weighted rules
  - there are various packages performing special modifications: rotating, longtable, the margintable environment in the tufte-handout document class
library(xtable)
fit = glm(Kyphosis ~ Age + I(Age^2) + Number + Start, 
          family = binomial, data = kyphosis)
xtable(summary(fit), caption = 
      "Logistic regression with kyphosis data set",
      label = "tab:example")

|            | Estimate | Std. Error | z value | Pr(>|z|) |
|------------|----------|------------|---------|----------|
| (Intercept)| -4.3836  | 2.0549     | -2.13   | 0.0329   |
| Age        | 0.0816   | 0.0345     | 2.36    | 0.0181   |
| I(Age^2)   | -0.0004  | 0.0002     | -2.08   | 0.0374   |
| Number     | 0.4269   | 0.2365     | 1.80    | 0.0711   |
| Start      | -0.2038  | 0.0707     | -2.88   | 0.0039   |

Table: Logistic regression with kyphosis data set
## Logistic regression with kyphosis data set

|            | Estimate | Std. Error | z value | Pr(>|z|) |
|------------|----------|------------|---------|----------|
| (Intercept)| -4.3836  | 2.0549     | -2.13   | 0.0329   |
| Age        | 0.0816   | 0.0345     | 2.36    | 0.0181   |
| I(Age^2)   | -0.0004  | 0.0002     | -2.08   | 0.0374   |
| Number     | 0.4269   | 0.2365     | 1.80    | 0.0711   |
| Start      | -0.2038  | 0.0707     | -2.88   | 0.0039   |
Problems

- The major problem is with the function `print.xtable()` which is nearly 700 lines of code.
- The code for producing HTML and for producing \LaTeX{} is bundled together in `print.xtable()`.
- The code in `print.xtable()` has been built up over a number of years, since version 1.0-1 in 2000.
- Logically similar code such as validation of input occurs at different places in `print.xtable()`.
Problems

- There are virtually no functions created or referenced within `print.xtable()`, except for `sanitize()`
- `print.xtable()` has a large argument list, of 32 arguments
- The main test suite consists of a vignette, *The xtable Gallery*, plus a vignette to illustrate the use of `margintable` and some test files I have written
- There are separate problems concerning the production of HTML code which I note but won’t be addressing today:
  - there is not much functionality for this
  - the HTML produced is not HTML5 compliant
Programming Principles

- **print.xtable()** violates most tenets of proper programming practice
  
  - **functions** Should not be overly long, and should perform a limited set of operations
  
  - **abstraction** Lower level details should be hidden away within functions
  
  - **modularity** Programs should be composed of distinct modules with specific functionality
  
  - **validation** User input should be validated and understandable error messages returned
  
  - **testing** Programs should be developed in tandem with testing procedures
Problems

- None of this should be taken as a criticism of David Dahl and other contributors to xtable.
- I am pretty sure most people who have written programs of any substantial size would do it differently if they were too start again on the same problem.
- Some illustrious R programmers have been known to release new versions of their packages with 2 appended to the original name.
Two approaches are possible to deal with the problems outlined:

- start again and produce `xtable2`
- refactor the package, that is reorganise and restructure the internals of `print.xtable()`

I am going to consider the latter approach today, without precluding possibly rewriting the package at some future time

Refactoring is useful even if the package is to be rewritten
<table>
<thead>
<tr>
<th>Line Numbers</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-66</td>
<td>object and attributes (obtained from user arguments in xtable and print.xtable functions)</td>
</tr>
<tr>
<td>60-66</td>
<td>assign captions</td>
</tr>
<tr>
<td>85-118</td>
<td>validation</td>
</tr>
<tr>
<td>120-154</td>
<td>create line rules (booktab dependent)</td>
</tr>
<tr>
<td>156-187</td>
<td>line rule locations</td>
</tr>
<tr>
<td>189-213</td>
<td>validation of user inputs</td>
</tr>
<tr>
<td>215-402</td>
<td>create LaTeX 'components'</td>
</tr>
<tr>
<td>366-402</td>
<td>LaTeX sanitize functions created</td>
</tr>
<tr>
<td>403-464</td>
<td>create HTML 'components'</td>
</tr>
<tr>
<td>437-463</td>
<td>HTML sanitize functions created</td>
</tr>
<tr>
<td>466-467</td>
<td>start recording table to 'result'</td>
</tr>
<tr>
<td>470-478</td>
<td>R.info and timestamp</td>
</tr>
<tr>
<td>480-496</td>
<td>caption, labels, size, tabular added</td>
</tr>
<tr>
<td>499-503</td>
<td>include rownames, colnames</td>
</tr>
<tr>
<td>504-532</td>
<td>sanitize, rotation for rownames and colnames</td>
</tr>
<tr>
<td>546-596</td>
<td>format digits</td>
</tr>
<tr>
<td>600-612</td>
<td>sanitize table content, apply NA string</td>
</tr>
<tr>
<td>614-624</td>
<td>create matrix to hold 'components'</td>
</tr>
<tr>
<td>627</td>
<td>combine matrix of 'components' with results</td>
</tr>
<tr>
<td>628-664</td>
<td>final latex components, add captions, etc.</td>
</tr>
<tr>
<td>665</td>
<td>final sanitize</td>
</tr>
<tr>
<td>667-672</td>
<td>return result</td>
</tr>
</tbody>
</table>
Original Structure of `print.xtable()`
First Stage of Refactoring

Refactoring the xtable Package

David J Scott, Daniel Geals, Paul Murrell
More Abstraction

- At this point `print.xtable()` is a sequence of function calls and subsequent assignments of variables returned from these function calls.
- To continue refactoring the function was divided into three sections:
  - pre-processing, involving validity checking and assignment of user arguments
  - a large `if-else` statement creating variables used for the final table
  - construction of the final table using the variables and pre-processed user arguments
print.xtable() Process

pre-process
(validation+ setup advanced user args)

buildParams
(create CAPS variables)

makeTable
(make final result)
Final Structure
Testing

- Two approaches were used to test the refactored code
- The .tex file produced by the vignette *The xtable Gallery* using the refactored code was compared to that produced by the original code, using `diff`
- Test functions were created using the `testthat` framework
To Do

- **xtable** has actually been updated since this work was done by Daniel Geals, so some updating and testing of the refactored code is required.

- Refactored code needs to be tested on packages depending on **xtable**.