Refactoring the xtable Package

David J Scott[1] Daniel Geals[1] Paul Murrell[1]

[1]Department of Statistics, The University of Auckland

July 10, 2015

Outline

Outline

Outline

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

Example

```
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

Example

```
## % latex table generated in R 3.1.2 by xtable 1.8-0 package
## % Fri Jul 10 15:00:01 2015
## \begin{table}[ht]
## \centering
## \begin{tabular}{rrrrr}
##   \hline
##  & Estimate & Std. Error & z value & Pr($>$$|$z$|$) \\
##   \hline
## (Intercept) & -4.3836 & 2.0549 & -2.13 & 0.0329 \\
##   Age & 0.0816 & 0.0345 & 2.36 & 0.0181 \\
##   I(Age\verb|^|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 \\
##    \hline
## \end{tabular}
## \caption{Logistic regression with kyphosis data set}
## \label{tab:example}
## \end{table}
```

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

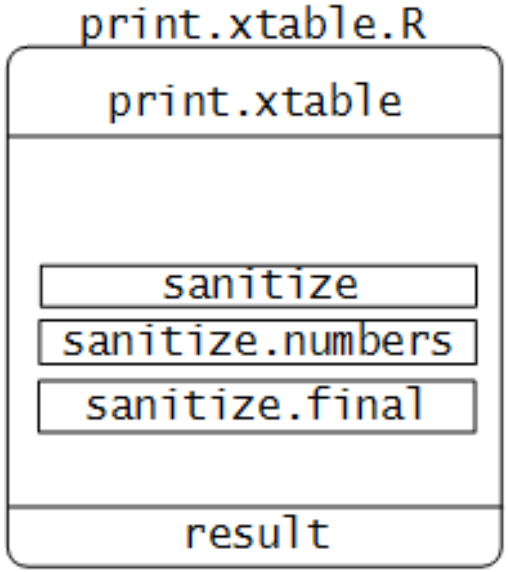
Problems

- 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

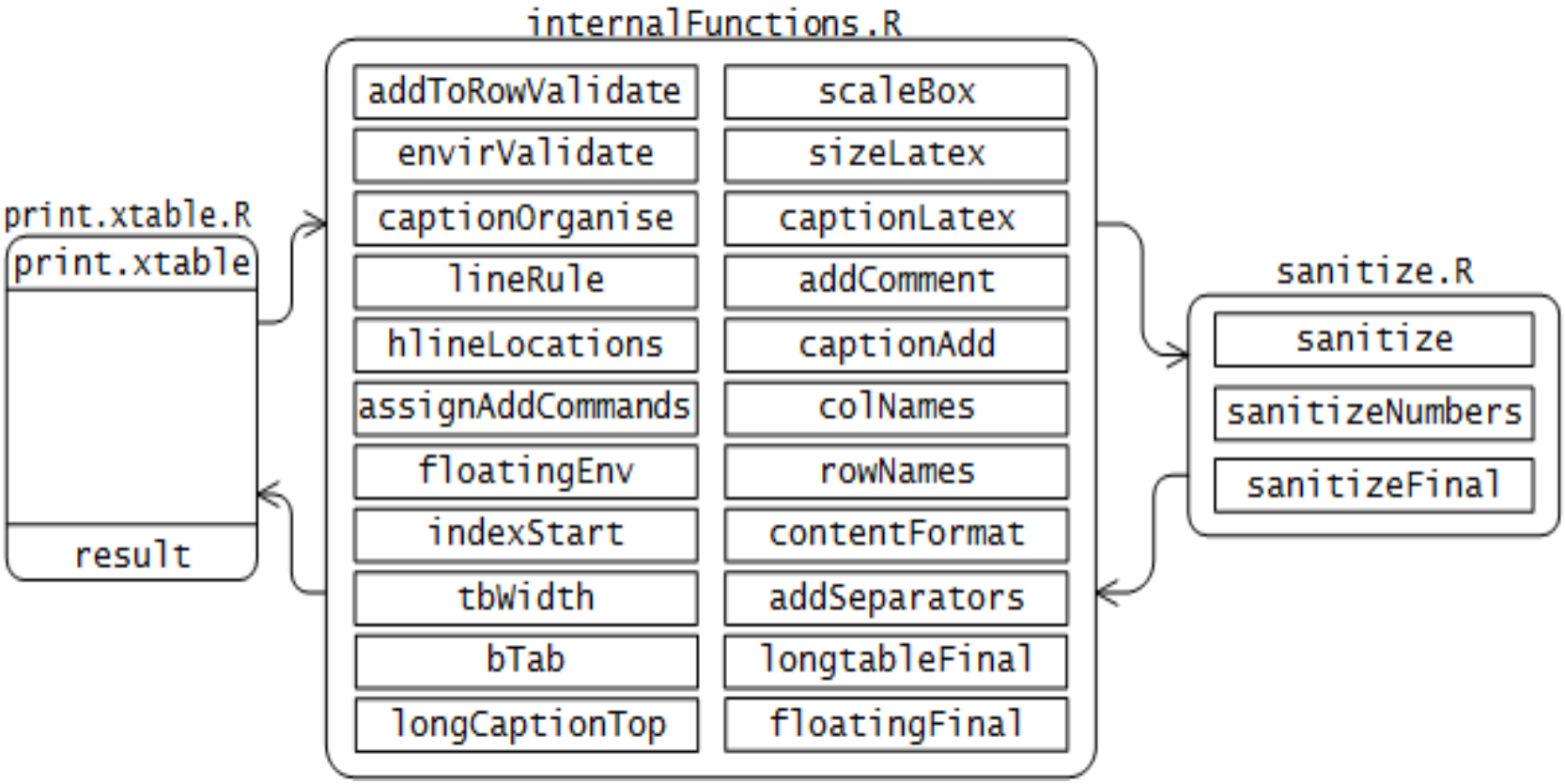
Pseudo-code Representation of `print.xtable()`

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

Original Structure of `print.xtable()`



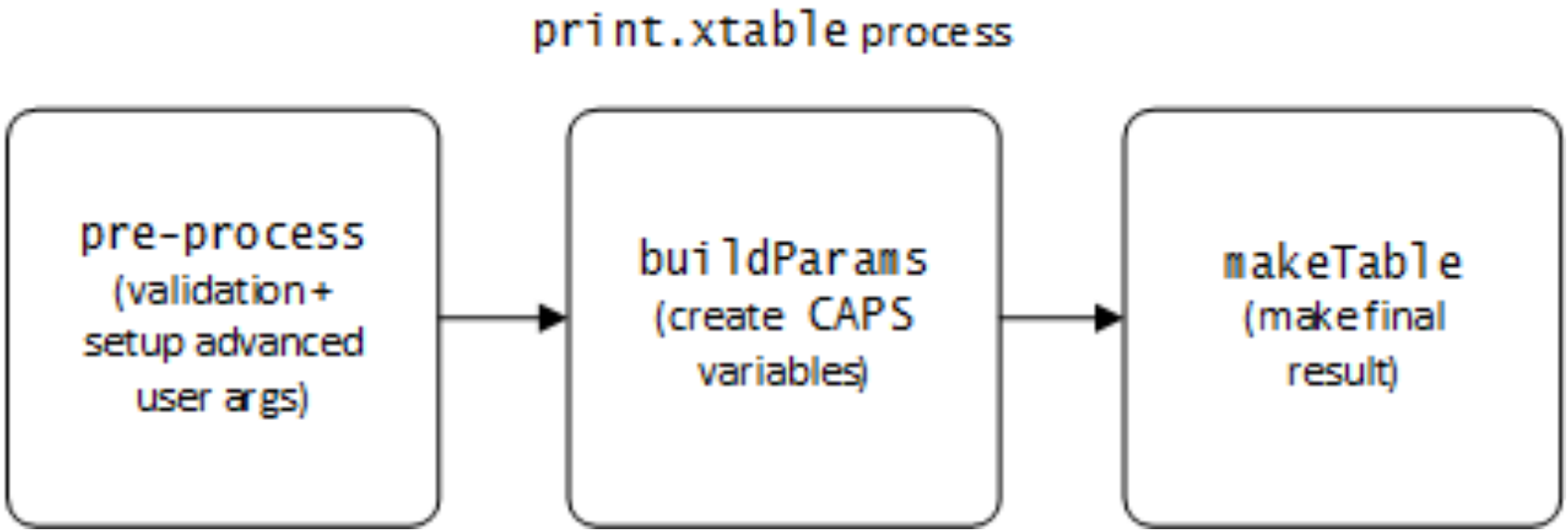
First Stage of Refactoring



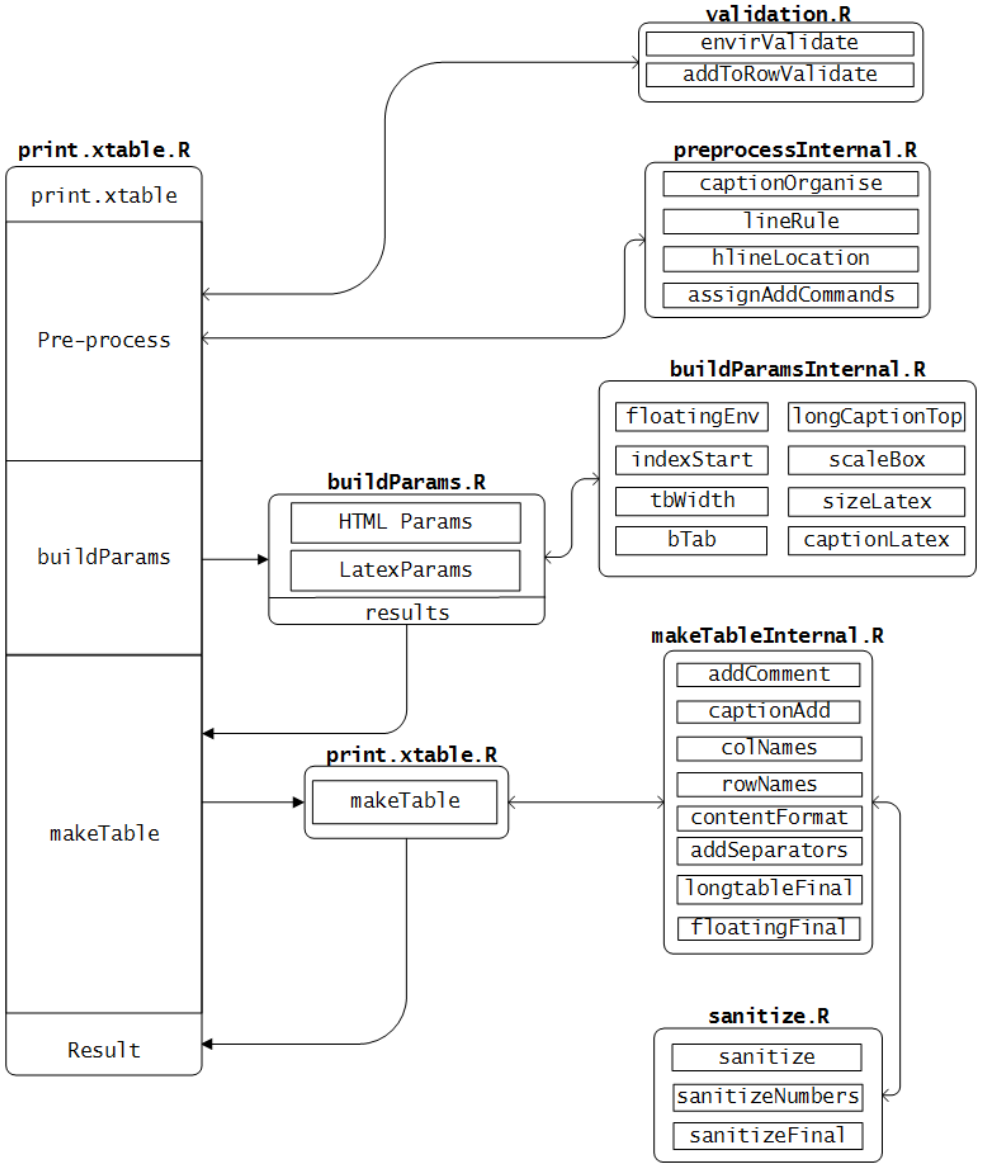
More Abstraction

- 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



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