Parsing Rd files

Duncan Murdoch

October 2, 2008

Abstract

R-devel (to become R 2.9.x) will soon have a parser for Rd format help files. This will allow easier processing, easier syntax checking, and eventually, easier conversion to other formats.

To write this parser, it has been necessary to make some small changes to the specification of the format as described in Writing R Extensions, and to make some choices when that description was ambiguous. Some existing Rd files do not meet the stricter format requirements.

This document describes the new format, necessary changes to existing Rd files, and the structure returned from the new `parseRd()` function.

1 Introduction

The R documentation (Rd) file format has not had a formal description. It has grown over time, with pieces added, and R and Perl code written to recognize it.

This document describes a formal parser (written in Bison) for the format, and the R structure that it outputs. The intention is to make it easier for the format to evolve, or to be replaced with a completely new one.

The next section describes the syntax; section 3 describes changes from what is described in Writing R Extensions for R 2.8.0. Finally, the last section describes the `parseRd()` function and its output. This document does not describe the interpretation of any of the markup in Rd files.

2 Rd Syntax Specification

Rd files are text files with an associated encoding, readable as text connections in R. The syntax only depends on a few ASCII characters, so at the level of this specification the encoding is not important, however higher level interpretation will depend on the text connection having read the proper encoding of the file.

There are three distinct types of text within an Rd file: LaTeX-like, R-like, and verbatim. The first two contain markup, text and comments; verbatim text contains only text.

2.1 Markup

Markup includes macros starting with a backslash `\`. Some macros (e.g. `\R`) take no arguments, some (e.g. `\code{}`) take one argument surrounded by braces, and some (e.g. `\section{}{}`) take two arguments surrounded by braces.

There are also three special classes of macros. The `\link{}` macro may take one argument or two, with the first marked as an option in square brackets, e.g. `\link[option]{arg}`. The `\eqn{}` and `deqn{}` macros may take one or two arguments in braces.

The last special class of macros consists of `#ifdef ... #endif` and `#ifndef ... #endif`. These look like C preprocessor directives, but are processed by the parser as two argument macros, with the first argument being the token on the line of the first directive, and the second one being the text between the first directive and the `#endif`. For example,

```
#ifdef unix
Some text
#endif
```

is processed with first argument `unix` and second argument `Some text`.

Markup macros are subdivided into those that start sections of the Rd file, and those that may be used within a section. The sectioning macros may only be used at the top level, while the others may be nested within the arguments of other macros.

Markup macros are also subdivided into those that contain list-like content, and those that don't. The `\item` macro may only be used within the argument of a list-like macro. Within `\enumerate{}` or

`\itemize{}` it takes no arguments, within `\arguments{}`, `\value{}` and `\describe{}` it takes two arguments.

Finally, the text within the argument of each macro is classified into one of the three types of text mentioned above. For example, the text within `\code{}` macros is R-like, and that within `\preformatted{}` macros is taken verbatim.

The complete list of Rd file macros is shown in Table **??**.

2.2 LaTeX-like text

LaTeX-like text in an Rd file is a stream of markup, text, and comments.

Comments in LaTeX-like mode start at a `%` character and run to the end of the line.

Text in LaTeX-like mode consists of all characters other than markup and comments. The white space within text is not considered relevant; currently the parser removes leading whitespace but may keep following whitespace. This is subject to change.

Braces within LaTeX-like text must be escaped as `\{` and `\}` except when used to delimit arguments to markup macros.

Brackets `[` and `]` within LaTeX-like text only have syntactic relevance after the `\link` macro, where they are used to delimit the optional argument.

Quotes `"`, `'` and `` ` `` have no syntactic relevance within LaTeX-like text.

2.3 R-like text

R-like text in an Rd file is a stream of markup, R code, and comments. The underlying mental model is that the markup could be replaced by suitable text and the R code would be parseable.

Comments in R-like mode start with `#` and run to the end of the line, or a brace that closes the block containing the R-like text. Unlike LaTeX-like comments, the Rd parser will return R comments as part of the text.

Quoted strings (using `"`, `'` or `` ` ``) within R-like text follow R rules: the string delimiters must match and markup and comments within them are taken to be part of the strings and are not interpreted by the Rd parser.

Braces within R-like text outside of quoted strings must balance, or be escaped.

Outside of a quoted string, in R-like text the escape character `\` indicates the start of a markup macro.

2.4 Verbatim text

Verbatim text within an Rd file is a pure stream of text, uninterpreted by the parser, with the exception that braces must balance or be escaped.

The escape character `\` is returned as is except in the two cases `\{` and `\}`, where it is deleted and the brace is included in the text.

3 Changes from R 2.8.x Rd format

The following list describes syntax that was accepted in R 2.8.x but which is no longer accepted by the parser.

1. Balanced braces within LaTeX-like text were ignored; now they are not allowed.
2. The `\R` macro was sometimes used with a single empty argument; now it must be used with no argument.
3. The `\code{}` macro was used for general verbatim text, similar to `\preformatted{}`; now the latter must be used when the text is not R-like. This mainly affects descriptions of escape sequences (e.g. `\code{\a}` must now be written as `\preformatted{\a}`, as otherwise `\a` would be taken to be a markup macro), and descriptions of languages other than R (e.g. examples of regular expressions are often not syntactically valid R).
4. Within verbatim text, only braces should be escaped.
5. Treating `#ifdef ... #endif` and `#ifndef ... #endif` as markup macros means that they must be wholly nested within other macros. For example, the construction

   ```
   \title{
   #ifdef unix
   Unix title}
   #endif
   #ifdef windows
   ```

```
Windows title}
#endif
```

needs to be rewritten as

```
\title{
#ifdef unix
Unix title
#endif
#ifdef windows
Windows title
#endif
}
```

6. R strings must be completely nested within markup macros. For example, `\code{"my string}"` will now be taken to be an unterminated `\code` macro, since the closing brace is within the string.

4 The parseRd() function

The `parseRd()` function takes a text connection and produces a parsed version of it. The general structure of the result is a list of section markup, with one entry per Rd section. Each section consists of a list of text and markup macros, with the text stored as one-element character vectors and the markup macros as lists.

The attributes of each element give information about the type of element. Details of this are still to be worked out.