The `jsonlite` Package: A Practical and Consistent Mapping Between JSON Data and R Objects

Jeroen Ooms
UCLA Department of Statistics

Abstract

A naive realization of `JSON` data in R maps `JSON` *arrays* to an `unnamed list`, and `JSON` *objects* to a `named list`. However, in practice a list is an awkward, inefficient type to store and manipulate data. Most statistical applications work with (homogeneous) vectors, matrices or data frames. Therefore `JSON` packages in R typically define certain special cases of `JSON` structures which map to simpler R types. Currently there exist no formal guidelines, or even consensus between implementations on how R data should be represented in `JSON`. Furthermore, upon closer inspection, even the most basic data structures in R actually do not perfectly map to their `JSON` counterparts and leave some ambiguity for edge cases. These problems have resulted in different behavior between implementations and can lead to unexpected output. This paper explicitly describes a mapping between R classes and `JSON` data, highlights potential problems, and proposes conventions that generalize the mapping to cover all common structures. We emphasize the importance of type consistency when using `JSON` to exchange dynamic data, and illustrate using examples and anecdotes. The `jsonlite` R package is used throughout the paper as a reference implementation.

1 Introduction

JavaScript Object Notation (`JSON`) is a text format for the serialization of structured data (Crockford, 2006a). It is derived from the object literals of `JavaScript`, as defined in the `ECMAScript` Programming Language Standard, Third Edition (ECMA, 1999). Design of `JSON` is simple and concise in comparison with other text based formats, and it was originally proposed by Douglas Crockford as a "fat-free alternative to `XML`" (Crockford, 2006b). The syntax is easy for humans to read and write, easy for machines to parse and generate and completely described in a single page at `http://www.json.org`. The character encoding of `JSON` text is always Unicode, using `UTF-8` by default (Crockford, 2006a), making it naturally compatible with non-latin alphabets. Over the past years, `JSON` has become hugely popular on the internet as a general purpose data interchange format. High quality parsing libraries are available for almost any programming language, making it easy to implement systems and applications that exchange data over the network using JSON. For R (R Core Team, 2013), several packages that assist the user in generating, parsing and validating JSON are available through CRAN, including `rjson` (Couture-Beil, 2013), `RJSONIO` (Lang, 2013), and `jsonlite` (Ooms et al., 2014).

The emphasis of this paper is not on discussing the `JSON` format or any particular implementation for using