Hard Test Analysis

Introduction

The hard test pushes the **boundaries** of XML parsing with the xml2 package, focusing on the **conversion** of an XML document into a **structured R list**. This test not only tests the package's parsing capabilities but also its ability to **transform** XML data into a format that is easily manipulable within R. It involves creating a **custom function** to recursively parse the XML document, demonstrating the xml2 package's flexibility and power in handling complex XML structures. The code snippet in this section provides a comprehensive example of how to leverage the xml2 package to parse XML documents into R lists, showcasing the package's robustness and versatility in XML data manipulation.

Section 1: Loading Libraries and XML Content

```
library(xml2)
library(stringr)
library(rlist)

z <- '
<CATALOG>
 <CD>
    <TITLE>Empire Burlesque</TITLE>
    <ARTIST>Bob Dylan</ARTIST>
    <COUNTRY>USA</COUNTRY>
    <COMPANY>Columbia</COMPANY>
    <PRICE>10.90</PRICE>
    <YEAR>1985</YEAR>
 </CD>
 <CD>
    <TITLE>Hide your heart</TITLE>
    <ARTIST>Bonnie Tylor</ARTIST>
    <COUNTRY>UK</COUNTRY>
    <COMPANY>CBS Records</COMPANY>
    <PRICE>9.90</PRICE>
    <YEAR>1988</YEAR>
 </CD>
</CATALOG>'
```

Explanation

- The **xml2, stringr**, and **rlist** libraries are loaded to handle XML data and list manipulation in R.
- An XML content string representing a **catalog** of **CDs** is defined, including details like title, artist, country, company, price, and year.

Section 2: Parsing XML to List Using rlist

```
res <- rlist::list.parse(z, type='xml')
res
```

```
## $CD
## $CD$TITLE
## [1] "Empire Burlesque"
##
## $CD$ARTIST
## [1] "Bob Dylan"
##
## $CD$COUNTRY
## [1] "USA"
##
## $CD$COMPANY
## [1] "Columbia"
##
## $CD$PRICE
## [1] "10.90"
##
## $CD$YEAR
## [1] "1985"
##
##
## $CD
## $CD$TITLE
## [1] "Hide your heart"
##
## $CD$ARTIST
## [1] "Bonnie Tylor"
##
## $CD$COUNTRY
## [1] "UK"
##
## $CD$COMPANY
## [1] "CBS Records"
##
## $CD$PRICE
## [1] "9.90"
##
## $CD$YEAR
## [1] "1988"
```

Explanation

- The **list.parse** function from the rlist package is used to parse the XML string into an R list.
- The **type='xml'** argument specifies that the input is XML content.
- The **parsed list** is stored in the variable res.

Section 3: Custom Function to Parse XML to List

```r
parse_xml_to_list <- function(xml_string) {
 xml_doc <- read_xml(xml_string)

 xml_to_list <- function(node) {

    if (xml_length(node) == 0) {
      return(xml_text(node))
    }

    else {
      children <- xml_children(node)
      list_result <- lapply(children, xml_to_list)
      return(setNames(list_result, xml_name(children)))
    }
 }

 result <- xml_to_list(xml_doc)

 return(result)
}
```

Explanation

- A custom function **parse_xml_to_list** is defined to parse an XML string into an R list.
- The function uses **recursion** to **traverse** the **XML document**. If a node has no children, it returns the text content of the node. Otherwise, it creates a list with the node's name as the key and the **children's list** as the value.
- The **xml_length** function is used to check if a node has children.
- The **xml_children** function is used to get the children of a node.
- The **xml_name** function is used to get the name of a node.
- The **xml_text** function is used to get the text content of a node.

Section 4: Using the Custom Function

```r
res2 <- parse_xml_to_list(z)

print(res2)
```

```
## $CD
## $CD$TITLE
## [1] "Empire Burlesque"
##
## $CD$ARTIST
## [1] "Bob Dylan"
##
## $CD$COUNTRY
## [1] "USA"
##
## $CD$COMPANY
## [1] "Columbia"
##
## $CD$PRICE
## [1] "10.90"
##
## $CD$YEAR
## [1] "1985"
##
##
## $CD
## $CD$TITLE
## [1] "Hide your heart"
##
## $CD$ARTIST
## [1] "Bonnie Tylor"
##
## $CD$COUNTRY
## [1] "UK"
##
## $CD$COMPANY
## [1] "CBS Records"
##
## $CD$PRICE
## [1] "9.90"
##
## $CD$YEAR
## [1] "1988"
```

```r
identical(res, res2)
```

```
## [1] TRUE
```

Explanation

- The custom function **parse_xml_to_list** is used to parse the XML string z into an R list, which is stored in res2.
- The **print** function is used to display the parsed list.
- The **identical** function checks if the list parsed by **rlist::list.parse** is identical to the list parsed by the custom function, demonstrating the equivalence of the two methods.