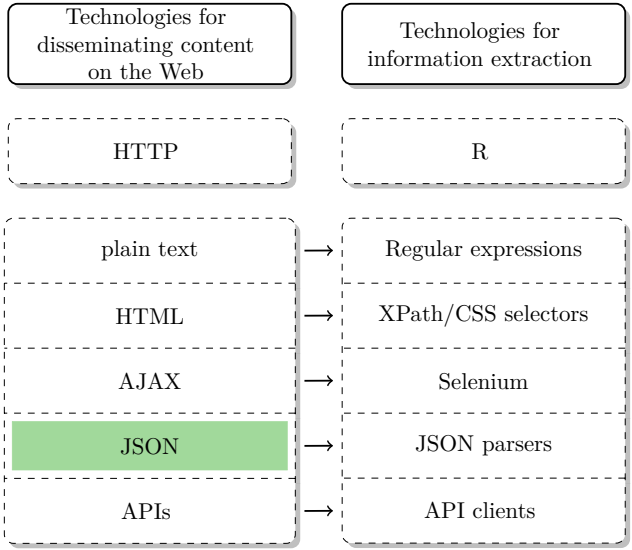
Introduction to Web Scraping with R

JSON

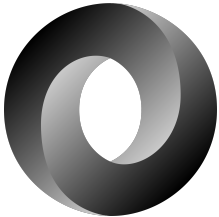
Simon Munzert | IPSDS

Technologies of the World Wide Web



What's JSON?

- **J**ava**S**cript **O**bject **N**otation
- popular data exchange format for web services / APIs
- 'the fat-free alternative to XML'
- JSON ≠ Java, but a subset of JavaScript
- however, very flexible and not dependent upon any programming language
- import of JSON data into R is relatively straightforward with the `jsonlite` package



Example

```
{"indy movies" :[
  {"name" : "Raiders of the Lost Ark",
  "year" : 1981,
        "actors" : {
               "Indiana Jones": "Harrison Ford",
               "Dr. Rene Belloq": "Paul Freeman"
               },
        "producers": ["Frank Marshall", "George Lucas", "Howard Kazanjian"],
        "budget" : 18000000,
        "academy_award_ve": true},
        {"name" : "Indiana Jones and the Temple of Doom",
        "year" : 1984,
        "actors" : {
               "Indiana Jones": "Harrison Ford",
               "Mola Ram": "Amish Puri"
               },
        "producers": ["Robert Watts"],
        "budget" : 28170000,
        "academy_award_ve": true}
  ]
}
```

Brackets

Types of brackets

1. curly brackets, '`{`' and '`}`', embrace **objects**. Objects work similar to elements in XML/HTML and can contain other objects, key-value pairs or arrays
2. square brackets, '`[`' and '`]`', embrace **arrays**. An array is an ordered sequence of objects or values.

Key-value pairs

1. Keys are put in quotation marks; values only if they contain string data.

```
"name" : "Indiana Jones and the Temple of Doom"
"year" : 1984
```

2. Keys and values are separated by a colon.

```
"year" : 1981
```

3. Key-value pairs are separated by commas.

```
{"Indiana Jones": "Harrison Ford",
"Dr. Rene Belloq": "Paul Freeman"}
```

4. Values within arrays are separated by commas.

```
["Frank Marshall", "George Lucas", "Howard Kazanjian"]
```

Data types

Summary

- JSON allows a basic set of data types
- often equivalent data types available in different programming languages
- compatibility with R partly given, but no isomophic translation possible

data type	meaning
number	integer, real, or floating point (e.g., 1.3E10)
string	whitespace, zero or more Unicode characters (except " or \; \ introduces some escape sequences)
boolean	`true` or `false`
null	`null`, an unknown value
Object	content in curly brackets
Array	ordered content in square brackets

JSON and R

Parsing software

- different packages available for R: `rjson`, `RJSONIO`, `jsonlite`
- choose `jsonlite`: it's under active development and provides convincing mapping rules

JSON and R

Parsing JSON with jsonlite

R code

```
1 (indy <- fromJSON("../materials/indy.json"))
```

```
$`indy movies`
                              name year actors.Indiana Jones
1             Raiders of the Lost Ark 1981        Harrison Ford
2 Indiana Jones and the Temple of Doom 1984        Harrison Ford
3   Indiana Jones and the Last Crusade 1989        Harrison Ford
  actors.Dr. Rene Belloq actors.Mola Ram actors.Walter Donovan
1          Paul Freeman            <NA>                  <NA>
2                  <NA>      Amish Puri                  <NA>
3                  <NA>            <NA>         Julian Glover
                                   producers   budget academy_award_ve
1 Frank Marshall, George Lucas, Howard Kazanjian 18000000             TRUE
2                                Robert Watts 28170000             TRUE
3                  Robert Watts, George Lucas 48000000            FALSE
```

end

JSON and R

Mapping rules of jsonlite

R code

```
library(jsonlite)
x <- '[1, 2, true, false]'
fromJSON(x)
```

```
[1] 1 2 1 0
```

```
x <- '["foo", true, false]'
fromJSON(x)
```

```
[1] "foo"   "TRUE"  "FALSE"
```

```
x <- '[1, "foo", null, false]'
fromJSON(x)
```

```
[1] "1"     "foo"   NA      "FALSE"
```

end

JSON and R

- there is no ultimate JSON-to-R converting function
- `jsonlite` simplifies matters a lot
- usually, JSON files returned by web services are not too complex