Handling JSON data

Sources:
G.Sanchez, H.
Wickham

```
{
  "Name": "Anakin",
  "Gender": "male",
  "Homeworld": "Tatooine",
  "Born": "41.9BBY",
  "Jedi": "yes"
},
{
  "Name": "Luke",
  "Gender": "male",
  "Homeworld": "Tatooine",
  "Born": "19BBY",
  "Jedi": "yes"
},
{
  "Name": "Leia",
  "Gender": "female",
  "Homeworld": "Alderaan",
  "Born": "19BBY",
  "Jedi": "no"
},
{
  "Name": "Obi-Wan",
```

JSON Data

Goal

JSON

The goal of these slides is to provide an introduction for **handling JSON data in R**

Synopsis

In a nutshell

We'll cover the following topics:

- JSON Basics
- R packages for JSON data
- Reading JSON data from the Web

Some References

- XML and Web Technlogies for Data Sciences with R
  by Deb Nolan and Duncan Temple Lang
- Introducing JSON
  http://www.json.org/
- R package RJSONIO
  http://cran.r-project.org/web/packages/RJSONIO/index.html
- R package jsonlite
  http://cran.r-project.org/web/packages/jsonlite/vignettes/json-mapping.pdf
- R package rjson
  http://cran.r-project.org/web/packages/rjson/index.html

JSON Basics

Basics First

Fundamentals

JSON stands for **JavaScript Object Notation** and it is a format for representing data

- general purpose format
- lightweight format
- widely popular
- fairly simple

Basics First

Why should we care?

When working with data from the Web, we'll inevitably find some JSON data

- JSON can be used directly in JavaScript code for Web pages
- many Web APIs provide data in JSON format
- R has packages designed to handle JSON data

Understanding JSON

Understanding JSON

JSON Data Types

- null
- true
- false
- number
- string

JSON Data Containers

- square brackets [ ]
- curly brackets { }

JSON Arrays

Unnamed Arrays

Square brackets [ ] are used for **ordered unnamed arrays**

- `[ 1, 2, 3, ... ]`
- `[ true, true, false, ... ]`

Named Arrays

Curly brackets { } are used for **named arrays**

- `{ "dollars" : 5, "euros" : 20, ... }`
- `{ "city" : "Berkeley", "state" : "CA", ... }`

JSON Arrays

Containers can be nested

Example A

```
{
    "name": ["X", "Y", "Z"],
    "grams": [300, 200, 500],
    "qty": [4, 5, null],
    "new": [true, false, true],
}
```

Example B

```
[
    { "name": "X",
      "grams": 300,
      "qty": 4,
      "new": true },
    { "name": "Y",
      "grams": 200,
      "qty": 5,
      "new": false },
    { "name": "Z",
      "grams": 500,
      "qty": null,
      "new": true}
]
```

Data Table Toy Example

Imagine we have some data

Name	Gender	Homeland	Born	Jedi
Anakin	male	Tatooine	41.9BBY	yes
Amidala	female	Naboo	46BBY	no
Luke	male	Tatooine	19BBY	yes
Leia	female	Alderaan	19BBY	no
Obi-Wan	male	Stewjon	57BBY	yes
Han	male	Corellia	29BBY	no
Palpatine	male	Naboo	82BBY	no
R2-D2	unknown	Naboo	33BBY	no

There are several ways to represent this data in JSON format

One way to represent data


```
[
    {
     "Name": "Anakin",
     "Gender": "male",
     "Homeworld": "Tatooine",
     "Born": "41.9BBY",
     "Jedi": "yes"
    },
    ...
    {
     "Name": "R2-D2",
     "Gender": "unknown",
     "Homeworld": "Naboo",
     "Born": "33BBY",
     "Jedi": "no"
    },
]
```

Another way to represent data


```
{
  "Name": [ "Anakin", "Amidala", "Luke", ... , "R2-D2" ],
  "Gender": [ "male", "female", "male", ... , "unknown" ],
  "Homeworld": [ "Tatooine", "Naboo", "Tatooine", ... , "Naboo" ],
  "Born": [ "41.9BBY", "46BBY", "19BBY", ... , "33BBY" ],
  "Jedi": [ "yes", "no", "yes", ... , "no" ]
}
```

JSON R packages

R packages

R packages for JSON

R has 3 packages for working with JSON data

- "RJSONIO" by Duncan Temple Lang
- "rjson" by Alex Couture-Beil
- "jsonlite" by Jeroen Ooms, Duncan Temple Lang, Jonathan Wallace

All packages provide 2 main functions —toJSON() and fromJSON()— that allow conversion **to** and **from** data in JSON format, respectively.

We'll focus on the functions from "RJSONIO"

R package RJSONIO

R package "RJSONIO"

If you don't have "RJSONIO" you'll have to install it:

```
# install RJSONIO
install.packages("RJSONIO", dependencies = TRUE)
```

R package RJSONIO

Main functions

There are 2 primary functions in "RJSONIO"

- toJSON() converts an R object to a string in JSON
- fromJSON() converts JSON content to R objects

Function toJSON()

```
toJSON(x, container = isContainer(x, asIs, .level),
       collapse = "\n", ...)
```

- `x` the R object to be converted to JSON format
- `container` whether to treat the object as a vector/container or a scalar
- `collapse` string used as separator when combining the individual lines of the generated JSON content
- `...` additional arguments controlling the JSON formatting

Function fromJSON()

```
fromJSON(content, handler = NULL, default.size = 100,
         depth = 150L, allowComments = TRUE, ...)
```

- `content` the JSON content: either a file name or a character string
- `handler` R object responsible for processing each individual token/element
- `deafult.size` size to use for arrays and objects in an effort to avoid reallocating each time we add a new element.
- `depth` maximum number of nested JSON levels
- `allowComments` whether to allow C-style comments within the JSON content
- `...` additional parameters

Data Table Toy Example

Imagine we have some tabular data

Name	Gender	Homeland	Born	Jedi
Anakin	male	Tatooine	41.9BBY	yes
Amidala	female	Naboo	46BBY	no
Luke	male	Tatooine	19BBY	yes
Leia	female	Alderaan	19BBY	no
Obi-Wan	male	Stewjon	57BBY	yes
Han	male	Corellia	29BBY	no
Palpatine	male	Naboo	82BBY	no
R2-D2	unknown	Naboo	33BBY	no

R Data Frame

```
# toy data
sw_data = rbind(
  c("Anakin", "male", "Tatooine", "41.9BBY", "yes"),
  c("Amidala", "female", "Naboo", "46BBY", "no"),
  c("Luke", "male", "Tatooine", "19BBY", "yes"),
  c("Leia", "female", "Alderaan", "19BBY", "no"),
  c("Obi-Wan", "male", "Stewjon", "57BBY","yes"),
  c("Han", "male", "Corellia", "29BBY", "no"),
  c("Palpatine", "male", "Naboo", "82BBY","no"),
  c("R2-D2", "unknown","Naboo", "33BBY", "no"))

# convert to data.frame and add column names
swdf = data.frame(sw_data)
names(swdf) = c("Name", "Gender", "Homeworld","Born",  "Jedi")
swdf
```

```
##          Name   Gender  Homeworld       Born Jedi
## 1      Anakin     male Tatooine 41.9BBY yes
## 2    Amidala   female      Naboo 46BBY no
## 3       Luke     male Tatooine  19BBY yes  Leia
## 4             female Alderaan  19BBY no Obi-Wan
## 5                       male Stewjon  57BBY yes
## 6        Han     male Corellia 29BBY no
## 7  Palpatine     male      Naboo    82BBY     no
## 8      R2-D2  unknown      Naboo    33BBY     no
```

From R to JSON

```
# load RJSONIO
library(RJSONIO)

# convert Rdata.frame to JSON
sw_json = toJSON(swdf)

# what class?
class(sw_json)


## [1] "character"


# display JSON format
cat(sw_json)


## {
## "Name": [ "Anakin", "Amidala", "Luke", "Leia", "Obi-Wan", "Han", "Palpatine", "R2-D2" ],
## "Gender": [ "male", "female", "male", "female", "male", "male", "male", "unknown" ],
## "Homeworld": [ "Tatooine", "Naboo", "Tatooine", "Alderaan", "Stewjon", "Corellia", "Naboo", "Naboo" ],
## "Born": [ "41.9BBY", "46BBY", "19BBY", "19BBY", "57BBY", "29BBY", "82BBY", "33BBY" ],
## "Jedi": [ "yes", "no", "yes", "no", "yes", "no", "no", "no" ]
## }
```

From JSON to R

```
# convert JSON string to R list
sw_R= fromJSON(sw_json)

# what class?
class(sw_R)


## [1] "list"


# display JSON format
sw_R


## $Name
##  [1] "Anakin"     "Amidala"    "Luke"       "Leia"       "Obi-Wan"    "Han"
## [7] "Palpatine" "R2-D2"
##
## $Gender
## [1] "male"       "female" "male"       "female" "male"       "male"      "male"
## [8] "unknown"
##
## $Homeworld
##  [1] "Tatooine"  "Naboo"      "Tatooine" "Alderaan" "Stewjon" "Corellia"
##  [7] "Naboo"      "Naboo"
##
## $Born
## [1] "41.9BBY" "46BBY" "19BBY" "19BBY" "57BBY" "29BBY"  "82BBY"
## [8] "33BBY"
##
## $Jedi
## [1] "yes" "no" "yes" "no" "yes" "no" "no"  "no"
```

Reading JSON Data

JSON Data from the Web

How do we read JSON data from the Web?

We read JSON data in several ways. One way is to pass the url directly to `fromJSON()`. Another way is by passing `fromJSON()` the name of the file with the JSON content as a single string.