

Tidying Hierarchical Data from the Web

Colin Rundel

2018-11-13



Data and Visualization
Services Department

Getting Started

R environments

We will be going through several live demos today and I will be making my code available as we go via a Dropbox link.

It will likely be most useful for you to follow along by taking this code as it is produced and copying and pasting it to your own R session.

- For this to work well please make sure that you are running a reasonable current version of R (3.5.1 is the latest).
- If you can't or don't want to update your local install, I highly recommend:
 - <https://vm-manage.oit.duke.edu/containers>

Packages

Everything we are going to use today is either part of or a dependency of the `tidyverse` metapackage. If you don't have that installed, now would be good to do so. (It is better to do it now since installation can take awhile)

```
install.packages("tidyverse")
```

Additionally, even if you do have it installed now is also a good time to check that all of your packages are up-to-date.

```
update.packages()
```

My sessionInfo

```
sessioninfo::package_info() %>% .[.$attached,]
```

```
##   package * version date     lib source
##  dplyr    * 0.7.8  2018-11-10 [1] CRAN (R 3.5.1)
##  jsonlite * 1.5   2017-06-01 [1] CRAN (R 3.5.0)
##  purrr    * 0.2.5  2018-05-29 [1] CRAN (R 3.5.0)
##  rvest    * 0.3.2  2016-06-17 [1] CRAN (R 3.5.0)
##  xml2     * 1.2.0  2018-01-24 [1] CRAN (R 3.5.0)
##
## [1] /usr/local/lib/R/3.5/site-library
## [2] /usr/local/Cellar/r/3.5.1/lib/R/library
```

Hierarchical Data?

Web data - html

```
<html>
  <body>

    <div>
      <span class="title">The Cat in the Hat</span>
      by <span class="author">Dr. Seuss</span>
      - <span class="price">$6.99</span>
    </div>

    <div>
      <span class="title">Edgar Gets Ready for Bed</span>
      by <span class="author">Jennifer Adams</span>,
        <span class="author">Ron Stucki</span>
      - <span class="price">$9.99</span>
    </div>

    <div>
      <span class="title">"More More More", Said the Baby</span>
      by <span class="author">Vera B Williams</span>
      - <span class="price">$7.19</span>
    </div>

  </body>
</html>
```

Web data - JSON

```
{  
  "bookList": [  
    {  
      "title": "The Cat in the Hat",  
      "author": "Dr. Seuss",  
      "price": 6.99  
    },  
    {  
      "title": "Edgar Gets Ready for Bed",  
      "author": ["Jennifer Adams", "Ron Stucki"],  
      "price": 9.99  
    },  
    {  
      "title": "\"More More More!\", Said the Baby",  
      "author": "Vera B Williams",  
      "price": 7.19  
    }  
  ]  
}
```

Web data - XML

```
<bookList>
  <book>
    <title>The Cat in the Hat</title>
    <author>Dr. Seuss</author>
    <price>6.99</price>
  </book>
  <book>
    <title>Edgar Gets Ready for Bed</title>
    <author>Jennifer Adams and Ron Stucki</author>
    <price>9.99</price>
  </book>
  <book>
    <title>"More More More," Said the Baby</title>
    <author>Vera B Williams</author>
    <price>7.19</price>
  </book>
</bookList>
```

Tidy data

country	year	cases	population
Afghanistan	1990	745	1567071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	172006362
Brazil	2000	80488	174904898
China	1999	219258	1272915272
China	2000	21666	1280428583

variables

country	year	cases	population
Afghanistan	1990	745	1567071
Afghanistan	2000	2666	20595360
Brazil	1999	31737	172006362
Brazil	2000	80488	174904898
China	1999	219258	1272915272
China	2000	21666	1280428583

observations

From R4DS - tidy data chapter, J. Stat Soft Paper (2014)

Motivating Examples

Denny's

LA QUINTA
INN

GREAT RATE
QUEEN BED
COURTYARD POOL

Dark Sky API

The easiest, most advanced, weather API on the web.

TRY FOR FREE

Easy to Use



Weather Conditions



Advanced Data



Web Scraping

rvest



rvest is a tidyverse package that makes basic processing and manipulation of HTML data straight forward. It wraps core functionality from the `xml2` package with a more user friendly interface, as such it can also be used for processing XML data.

Core functions:

- `read_html`
- `html_nodes`
- `html_table`
- `html_text`
- `html_name`
- `htmlAttrs`
- `html_attr`

html + rvest

```
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p align="center">Hello world!</p>
    <br/>
    <div class="name" id="first">John</div>
    <div class="name" id="last">Doe</div>
    <div class="contact">
      <div class="home">555-555-1234</div>
      <div class="home">555-555-2345</div>
      <div class="work">555-555-9999</div>
      <div class="fax">555-555-8888</div>
    </div>
  </body>
</html>
```

```
read_html(html)
```

```
## {xml_document}
## <html>
## [1] <head>\n<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">\n<title>
## [2] <body>\n      <p align="center">Hello world!</p>\n      <br><div class="name" id="first">
```

css selectors

We will be using a tool called **Selector Gadget** to help identify and extract the html elements / attributes of interest - it does this by interactively constructing a css selector using an example html document.

Selector	Example	Description
element	p	Select all <p> elements
element element	div p	Select all <p> elements inside a <div> element
element>element	div > p	Select all <p> elements with <div> as a parent
.class	.title	Select all elements with class="title" attribute
#id	#name	Select all elements with id="name" attribute
[attribute]	[class]	Select all elements with a class attribute
[attribute=value]	[class=title]	Select all elements with class="title"

Selecting and Extracting

```
...  
<p align="center">Hello world!</p>  
...
```

```
read_html(html) %>% html_nodes("p")
```

```
## {xml_nodeset (1)}  
## [1] <p align="center">Hello world!</p>
```

Selecting and Extracting

```
...  
<p align="center">Hello world!</p>  
...
```

```
read_html(html) %>% html_nodes("p")
```

```
## {xml_nodeset (1)}  
## [1] <p align="center">Hello world!</p>
```

```
read_html(html) %>% html_nodes("p") %>% html_text()
```

```
## [1] "Hello world!"
```

Selecting and Extracting

```
...  
<p align="center">Hello world!</p>  
...
```

```
read_html(html) %>% html_nodes("p")
```

```
## {xml_nodeset (1)}  
## [1] <p align="center">Hello world!</p>
```

```
read_html(html) %>% html_nodes("p") %>% html_text()
```

```
## [1] "Hello world!"
```

```
read_html(html) %>% html_nodes("p") %>% html_attrs()
```

```
## [[1]]  
##   align  
##   "center"
```

Selecting and Extracting

```
...  
<p align="center">Hello world!</p>  
...
```

```
read_html(html) %>% html_nodes("p")
```

```
## {xml_nodeset (1)}  
## [1] <p align="center">Hello world!</p>
```

```
read_html(html) %>% html_nodes("p") %>% html_text()
```

```
## [1] "Hello world!"
```

```
read_html(html) %>% html_nodes("p") %>% html_attrs()
```

```
## [[1]]  
##   align  
##   "center"
```

```
read_html(html) %>% html_nodes("p") %>% html_attr("align")
```

```
## [1] "center"
```

Nesting

```
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p align="center">Hello world!</p>
    <br/>
    <div class="name" id="first">John</div>
    <div class="name" id="last">Doe</div>
    <div class="contact">
      <div class="home">555-555-1234</div>
      <div class="home">555-555-2345</div>
      <div class="work">555-555-9999</div>
      <div class="fax">555-555-8888</div>
    </div>
  </body>
</html>
```

```
read_html(html) %>% html_nodes("body div")
```

```
## [1] <div class="name" id="first">John</div>
## [2] <div class="name" id="last">Doe</div>
## [3] <div class="contact">\n      <div class="home">555-555-1234</div>\n      <div class="home">555-555-2345</div>\n      <div class="work">555-555-9999</div>\n      <div class="fax">555-555-8888</div>
```

Nesting

```
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p align="center">Hello world!</p>
    <br/>
    <div class="name" id="first">John</div>
    <div class="name" id="last">Doe</div>
    <div class="contact">
      <div class="home">555-555-1234</div>
      <div class="home">555-555-2345</div>
      <div class="work">555-555-9999</div>
      <div class="fax">555-555-8888</div>
    </div>
  </body>
</html>
```

```
read_html(html) %>% html_nodes("body>div")
```

```
## {xml_nodeset (3)}
## [1] <div class="name" id="first">John</div>
## [2] <div class="name" id="last">Doe</div>
## [3] <div class="contact">\n      <div class="home">555-555-1234</div>\n      <div class="home">555-555-2345</div>\n      <div class="work">555-555-9999</div>\n      <div class="fax">555-555-8888</div>\n    </div>
```

Nesting

```
<html>
  <head>
    <title>This is a title</title>
  </head>
  <body>
    <p align="center">Hello world!</p>
    <br/>
    <div class="name" id="first">John</div>
    <div class="name" id="last">Doe</div>
    <div class="contact">
      <div class="home">555-555-1234</div>
      <div class="home">555-555-2345</div>
      <div class="work">555-555-9999</div>
      <div class="fax">555-555-8888</div>
    </div>
  </body>
</html>
```

```
read_html(html) %>% html_nodes("body div div")
```

```
## {xml_nodeset (4)}
## [1] <div class="home">555-555-1234</div>
## [2] <div class="home">555-555-2345</div>
## [3] <div class="work">555-555-9999</div>
## [4] <div class="fax">555-555-8888</div>
```

classes and ids

```
read_html(html) %>% html_nodes(".name")  
  
## {xml_nodeset (2)}  
## [1] <div class="name" id="first">John</div>  
## [2] <div class="name" id="last">Doe</div>
```

classes and ids

```
read_html(html) %>% html_nodes(".name")
```

```
## {xml_nodeset (2)}
## [1] <div class="name" id="first">John</div>
## [2] <div class="name" id="last">Doe</div>
```

```
read_html(html) %>% html_nodes("div.name")
```

```
## {xml_nodeset (2)}
## [1] <div class="name" id="first">John</div>
## [2] <div class="name" id="last">Doe</div>
```

classes and ids

```
read_html(html) %>% html_nodes(".name")
```

```
## {xml_nodeset (2)}
## [1] <div class="name" id="first">John</div>
## [2] <div class="name" id="last">Doe</div>
```

```
read_html(html) %>% html_nodes("div.name")
```

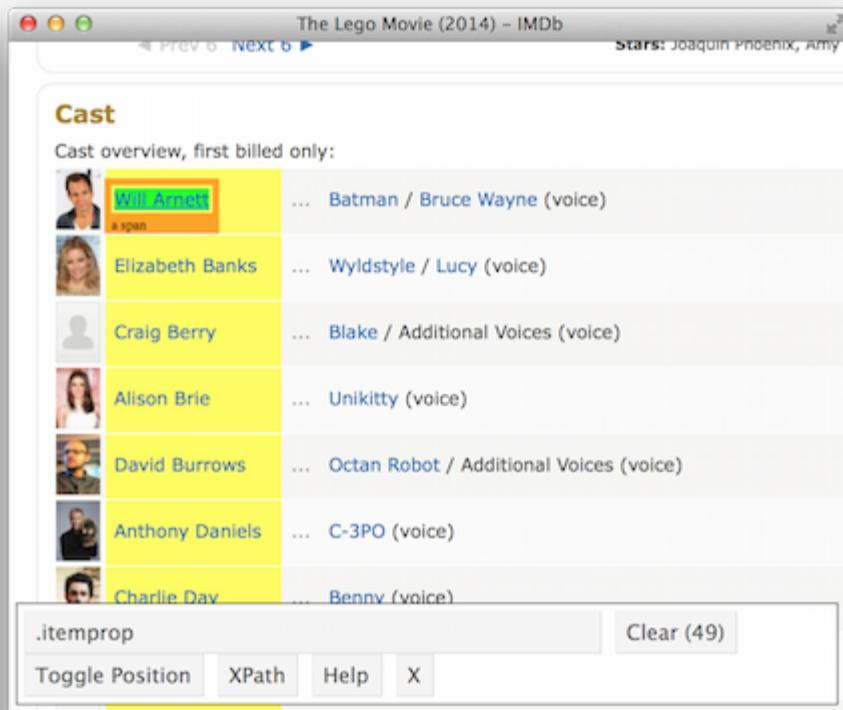
```
## {xml_nodeset (2)}
## [1] <div class="name" id="first">John</div>
## [2] <div class="name" id="last">Doe</div>
```

```
read_html(html) %>% html_nodes("#first")
```

```
## {xml_nodeset (1)}
## [1] <div class="name" id="first">John</div>
```

SelectorGadget

Is a javascript scriptlet / Chrome extension that helps you interactively build a CSS selector for the content you are interested in.



Live Demo

Putting rvest and Selector Gadget to work

http://bit.ly/Rundel-DVS_dennys

Dark Sky API

Dark Sky API Details

Go to <https://darksky.net/dev> and sign up for an account.

Every account gets up to 1000 free API calls / day which should be more than enough for our purposes today.

Dark Sky API Details

Go to <https://darksky.net/dev> and sign up for an account.

Every account gets up to 1000 free API calls / day which should be more than enough for our purposes today.

This API uses a **R**epresentational **S**tate **T**ransfer (REST) design, we can make two kinds of requests with this API.

- Forecast Request:

```
https://api.darksky.net/forecast/{key}/{latitude},{longitude}
```

- Time Machine Request:

```
https://api.darksky.net/forecast/{key}/{latitude},{longitude},{time}
```

Durham Forecast results

The latitude and longitude of Durham, NC are 35.9940° N, 78.8986° W which means we need to request

<https://api.darksky.net/forecast/24e13b35a014e3ca53a36a217243f61d/35.9940,-78.8986> from the API.

Durham Forecast results

The latitude and longitude of Durham, NC are 35.9940° N, 78.8986° W which means we need to request

<https://api.darksky.net/forecast/24e13b35a014e3ca53a36a217243f61d/35.9940,-78.8986> from the API.

The results look something like the following, where each \oplus represents hidden nested values.

```
{
  "latitude": 35.994,
  "longitude": -78.8986,
  "timezone": "America/New_York",
  "currently": {  $\oplus$  },
  "minutely": {  $\oplus$  },
  "hourly": {  $\oplus$  },
  "daily": {  $\oplus$  },
  "flags": {  $\oplus$  },
  "offset": -5
}
```

Note - We can check our coordinates using google maps:
<https://www.google.com/maps?q=35.9940,-78.8986>

jsonlite

To get things into R we can use the `jsonlite` package to directly read the url.

```
durham = jsonlite::fromJSON(  
  "https://api.darksky.net/forecast/24e13b35a014e3ca53a36a217243f61d/35.9940,-7  
  simplifyDataFrame = FALSE  
)
```

```
str(durham)
```

```
## List of 10  
## $ latitude : num 36  
## $ longitude: num -78.9  
## $ timezone : chr "America/New_York"  
## $ currently:List of 20  
##   ..$ time                  : int 1542085853  
##   ..$ summary                : chr "Rain"  
##   ..$ icon                  : chr "rain"  
##   ..$ nearestStormDistance: int 0  
##   ..$ precipIntensity       : num 0.12  
##   ..$ precipIntensityError: num 0.0299  
##   ..$ precipProbability     : int 1  
##   ..$ precipType             : chr "rain"  
##   ..$ temperature            : num 45.6  
##   ..$ apparentTemperature    : num 44.2  
##   ..$ dewPoint               : num 45.6  
##   ..$ humidity               : int 1
```

Complex lists - str

```
str(durham, max.level=1)
```

```
## List of 10
## $ latitude : num 36
## $ longitude: num -78.9
## $ timezone : chr "America/New_York"
## $ currently:List of 20
## $ minutely :List of 3
## $ hourly   :List of 3
## $ daily    :List of 3
## $ alerts   :List of 2
## $ flags    :List of 3
## $ offset   : int -5
```

```
str(durham$currently, max.level=1)
```

```
## List of 20
## $ time                  : int 1542085853
## $ summary                : chr "Rain"
## $ icon                  : chr "rain"
## $ nearestStormDistance: int 0
## $ precipIntensity       : num 0.12
## $ precipIntensityError: num 0.0299
## $ precipProbability     : int 1
## $ precipType              : chr "rain"
## $ temperature             : num 45.6
## $ apparentTemperature    : num 44.2
## $ dewPoint                : num 45.6
## $ humidity                 : int 1
## $ pressure                : num 1016
## $ windSpeed               : num 3.55
## $ windGust                : num 4.55
## $ windBearling            : int 338
## $ cloudCover              : int 1
## $ uvIndex                 : int 0
## $ visibility              : num 2.88
## $ ozone                   : num 230
```

Complex lists - View

View(durham)

Name	Type	Value
durham	list [9]	List of length 9
latitude	double [1]	35.994
longitude	double [1]	-78.8986
timezone	character [1]	'America/New_York'
currently	list [19]	List of length 19
minutely	list [3]	List of length 3
summary	character [1]	'Partly cloudy for the hour.'
icon	character [1]	'partly-cloudy-day'
data	list [61]	List of length 61
[[1]]	list [3]	List of length 3
time	integer [1]	1541972820
precipIntensity	integer [1]	0
precipProbability	integer [1]	0
[[2]]	list [3]	List of length 3
[[3]]	list [3]	List of length 3
durham		

Name	Type	Value
durham	list [9]	List of length 9
latitude	double [1]	35.994
longitude	double [1]	-78.8986
timezone	character [1]	'America/New_York'
currently	list [19]	List of length 19
minutely	list [3]	List of length 3
hourly	list [3]	List of length 3
summary	character [1]	'Rain tomorrow afternoon.'
icon	character [1]	'rain'
data	list [49]	List of length 49
[[1]]	list [17]	List of length 17
time	integer [1]	1541970000
summary	character [1]	'Partly Cloudy'
icon	character [1]	'partly-cloudy-day'
precipIntensity	integer [1]	0
precipProbability	integer [1]	0
temperature	double [1]	49.79
apparentTempe...	double [1]	49.79
dewPoint	double [1]	21.22
humidity	double [1]	0.32
pressure	double [1]	1027.8
windSpeed	double [1]	1.11
windGust	double [1]	5.81
durham["hourly"]		

purrr



purrr is a tidyverse package which improves the functional programming tools in R (i.e. lapply, sapply, etc.). It focuses on providing *pure* and *type stable* functions as well as several convenient shortcuts for common tasks.

purrr



purrr is a tidyverse package which improves the functional programming tools in R (i.e. lapply, sapply, etc.). It focuses on providing *pure* and *type stable* functions as well as several convenient shortcuts for common tasks.

Key functions:

- `map()`
- `map_lgl()`
- `map_int()`
- `map_dbl()`
- `map_chr()`
- `map_df() / map_dfr()`
- `map_dfc()`
- `walk()`

Type Consistency

R is a weakly / dynamically typed language which means there is no simple way to define a function which enforces the argument or return types of a function.

This flexibility can be useful at times, but often it makes it hard to reason about your code and requires more verbose code to handle edge cases.

```
x = list(1:3, 4:6, 7:9)
```

```
sapply(x, mean)
```

```
## [1] 2 5 8
```

```
lapply(x, mean)
```

```
## [[1]]  
## [1] 2  
##  
## [[2]]  
## [1] 5  
##  
## [[3]]  
## [1] 8
```

```
map_dbl(x, mean)
```

```
## [1] 2 5 8
```

```
map_chr(x, mean)
```

```
## [1] "2.000000" "5.000000" "8.000000"
```

```
map_int(x, mean)
```

```
## Error: Can't coerce element 1 from a double to a integer
```

Purrr shortcut - Lookups

Very often we want to extract only certain (named) values from a list, `purrr` provides a shortcut for this operation by providing either a character or numeric value instead of a function to the `map` function.

```
json = jsonlite::fromJSON(  
'{  
  "bookList": [  
    {"title": "The Cat in the Hat",  
     "author": "Dr. Seuss",  
     "price": 6.99},  
    {"title": "Edgar Gets Ready for Bed",  
     "author": ["Jennifer Adams", "Ron Stucki"],  
     "price": 9.99},  
    {"title": "\"More More More\"", Said the Baby,  
     "author": "Vera B Williams",  
     "price": 7.19}  
  ]  
}', simplifyVector = FALSE  
)
```

```
str(json)  
  
## List of 1  
## $ bookList:List of 3  
##   ..$ :List of 3  
##     ...$ title : chr "The Cat in the Hat"  
##     ...$ author: chr "Dr. Seuss"  
##     ...$ price : num 6.99  
##   ..$ :List of 3  
##     ...$ title : chr "Edgar Gets Ready for Bed"  
##     ...$ author:List of 2  
##       ...$ : chr "Jennifer Adams"  
##       ...$ : chr "Ron Stucki"  
##     ...$ price : num 9.99  
##   ..$ :List of 3  
##     ...$ title : chr "\"More More More\"", Said the Baby"  
##     ...$ author: chr "Vera B Williams"  
##     ...$ price : num 7.19
```

Purrr shortcut - Lookups

```
map_chr(json$bookList, "title")
```

```
## [1] "The Cat in the Hat"           "Edgar Gets Ready for Bed"  
## [3] "\"More More More\"", Said the Baby"
```

Purrr shortcut - Lookups

```
map_chr(json$bookList, "title")
```

```
## [1] "The Cat in the Hat"           "Edgar Gets Ready for Bed"  
## [3] "\"More More More\"", Said the Baby"
```

```
map_dbl(json$bookList, "price")
```

```
## [1] 6.99 9.99 7.19
```

Purrr shortcut - Lookups

```
map_chr(json$bookList, "title")
```

```
## [1] "The Cat in the Hat"           "Edgar Gets Ready for Bed"  
## [3] "\"More More More\"", Said the Baby"
```

```
map_dbl(json$bookList, "price")
```

```
## [1] 6.99 9.99 7.19
```

```
map_chr(json$bookList, "author")
```

```
## Error: Result 2 is not a length 1 atomic vector
```

Purrr shortcut - Lookups

```
map_chr(json$bookList, "title")
```

```
## [1] "The Cat in the Hat"           "Edgar Gets Ready for Bed"  
## [3] "\"More More More\"", Said the Baby"
```

```
map_dbl(json$bookList, "price")
```

```
## [1] 6.99 9.99 7.19
```

```
map_chr(json$bookList, "author")
```

```
## Error: Result 2 is not a length 1 atomic vector
```

```
map_chr(json$bookList, list("author", 1))
```

```
## [1] "Dr. Seuss"      "Jennifer Adams"  "Vera B Williams"
```

Live Demo

Putting purrr to work

http://bit.ly/Rundel-DVS_darksky