R Lab 4: Introduction to Data Extraction

Simon Caton

Getting data online

The easy way to get data is to go to some of the standard data repositories and download it, e.g.:

- http://dublinked.ie
- http://archive.ics.uci.edu/ml/index.php
- https://www.kaggle.com/datasets
- https://data.europa.eu/euodp/data/
- https://zenodo.org
- https://github.com/caesar0301/awesome-public-datasets
- etc.

Sometimes though these kinds of sources are not enough, we want to access dynamic datasets, often via application programming interfaces (APIs), or simply by scrapping data from the web.

In this part of the course we're going to attempt to introduce the following across labs 4 and 5: - Grabbing tables straight out of webpage - Pulling live as well as historical data from APIs - Accessing various forms of social media - Preparing data from these various sources for exploration similar to Labs 1-3 and perhaps something a little more interesting.

To try and keep a semi consistent story throughout we're going to explore data sources that are in some way related to a Game of Thrones.

Simple Structured Data from Static Web Pages

HBO have adapted the book series into a TV series, currently of 7 seasons. Conveniently, Wikipedia has a page on Game of Thrones https://en.wikipedia.org/wiki/Game_of_Thrones. So let's see what we can get out of this page.

```r
library(htmltab)
url <- "https://en.wikipedia.org/wiki/Game_of_Thrones"
seasons <-  htmltab(doc=url, which=2)
```

```
## Warning: Columns [Refs] seem to have no data and are removed. Use
## rm_nodata_cols = F to suppress this behavior
```

```r
head(seasons, 7)
```

```
##     Season        Ordered                    Filming    First aired
## 2 Season 1  March 2, 2010         Second half of 2010 April 17, 2011
## 3 Season 2 April 19, 2011         Second half of 2011  April 1, 2012
## 4 Season 3 April 10, 2012        July - November 2012 March 31, 2013
## 5 Season 4  April 2, 2013        July - November 2013  April 6, 2014
## 6 Season 5  April 8, 2014        July - December 2014 April 12, 2015
## 7 Season 6  April 8, 2014        July - December 2015 April 24, 2016
## 8 Season 7 April 21, 2016 August 2016 - February 2017  July 16, 2017
##         Last aired
## 2   June 19, 2011
## 3    June 3, 2012
## 4    June 9, 2013
## 5   June 15, 2014
## 6   June 14, 2015
## 7   June 26, 2016
## 8 August 27, 2017
##
## 2
## 3                                                                                  A Clash of King
## 4
## 5                                     The remaining one-third of A Storm of Swords and some elements
## 6 A Feast for Crows, A Dance with Dragons and original content, with some late chapters from A Storm
## 7                      Original content and outline from The Winds of Winter, with some late elements
## 8                                                                            Original content and outlin
```

The parameter *which* is used to specify the table of interest. Check the page, we've extracted the second table in the article. The first table is top right and provides some basic info about Game of Thrones in general. The warning message pertains to the refs column; the package is not pulling hyperlinks within the table, so this column appears empty. Note also the information loss on the links within the table that refer to other wikipedia articles.

Note that the row names are wrong! So let's fix them.

```r
rownames(seasons) <- c(1:8)
head(seasons)
```

```
##     Season        Ordered              Filming    First aired
## 1 Season 1  March 2, 2010  Second half of 2010 April 17, 2011
## 2 Season 2 April 19, 2011  Second half of 2011  April 1, 2012
## 3 Season 3 April 10, 2012 July - November 2012 March 31, 2013
## 4 Season 4  April 2, 2013 July - November 2013  April 6, 2014
## 5 Season 5  April 8, 2014 July - December 2014 April 12, 2015
## 6 Season 6  April 8, 2014 July - December 2015 April 24, 2016
##      Last aired
## 1 June 19, 2011
## 2  June 3, 2012
## 3  June 9, 2013
## 4 June 15, 2014
## 5 June 14, 2015
## 6 June 26, 2016
##
## 1
## 2                                                                                  A Clash of King
## 3
## 4                                     The remaining one-third of A Storm of Swords and some elements
## 5 A Feast for Crows, A Dance with Dragons and original content, with some late chapters from A Storm
## 6                      Original content and outline from The Winds of Winter, with some late elements
```

That looks better.

Conveniently, the article also has some additional data on reviewer scores for these seasons. Let's get those too:

```r
criticResponses <-  htmltab(doc=url, which=3)
```

```
## Warning: Columns [Season] seem to have no data and are removed. Use
## rm_nodata_cols = F to suppress this behavior
```

```r
head(criticResponses)
```

```
##   Season Critical response >> Rotten Tomatoes
## 3      1                    89% (33 reviews)
## 4      2                    96% (33 reviews)
## 5      3                    97% (44 reviews)
## 6      4                    97% (50 reviews)
## 7      5                    95% (52 reviews)
## 8      6                    96% (29 reviews)
##   Critical response >> Metacritic
## 3                 80 (28 reviews)
## 4                 90 (26 reviews)
## 5                 91 (25 reviews)
## 6                 94 (29 reviews)
## 7                 91 (29 reviews)
## 8                  73 (9 reviews)
```

Again, we need to fix the row names. Interestingly, the 8th season is available on Wikipedia, but there are no reviewer scores for it yet. If this is still the case, when you run the lab, the merge below will kick out season 8.

Tasks

1. Fix the row names
2. Encode the Season attibute in both data.frames such that both are factors, and such that they **both** use the same level labels. We need them to be represented in the same way, as we are going to use this column as a reference point to merge the two tables into one table.

Output to task 2, should look something like:

```
##  Factor w/ 8 levels "1","2","3","4",..: 1 2 3 4 5 6 7 8
```

Or if you prefer, you can also prefix it with the word Season, it's really up to you. Key, however, is that they are the same!

Now, let's merge the two data.frames into one.

```
GoTWikipedia <- merge(seasons, criticResponses, by="Season")
head(GoTWikipedia)
```

```
##   Season        Ordered              Filming    First aired    Last aired
## 1      1  March 2, 2010  Second half of 2010 April 17, 2011 June 19, 2011
## 2      2 April 19, 2011  Second half of 2011  April 1, 2012  June 3, 2012
## 3      3 April 10, 2012 July - November 2012 March 31, 2013  June 9, 2013
## 4      4  April 2, 2013 July - November 2013  April 6, 2014 June 15, 2014
## 5      5  April 8, 2014 July - December 2014 April 12, 2015 June 14, 2015
## 6      6  April 8, 2014 July - December 2015 April 24, 2016 June 26, 2016
##
## 1
## 2                                                                       A Clash of King
## 3
## 4                                     The remaining one-third of A Storm of Swords and some elements
## 5 A Feast for Crows, A Dance with Dragons and original content, with some late chapters from A Storm
## 6                     Original content and outline from The Winds of Winter, with some late elements
##   Critical response >> Rotten Tomatoes Critical response >> Metacritic
## 1                     89% (33 reviews)                 80 (28 reviews)
## 2                     96% (33 reviews)                 90 (26 reviews)
## 3                     97% (44 reviews)                 91 (25 reviews)
## 4                      97% (50 reviews)                 94 (29 reviews)
## 5                      95% (52 reviews)                 91 (29 reviews)
## 6                      96% (29 reviews)                  73 (9 reviews)
```

Optional Digression

Just to deal with the pendantic question of, but they are in the same order and have the same number of rows, I could have just stuck one on the end of the other. Let's demo what we just did by adding some noise to our data.

We're going to make some fake seasons, add random review scores to them, and some that the merge is the intersection of the two data.frames and not their union.

```
fakeSeasons <- c(8:100)

#if we were using the original representation of this field instead:
#fakeSeasons <- paste0("Season ", c(8:100))
fakeText <- rep("fake", length(fakeSeasons))

fake.df.seasons <- data.frame(fakeSeasons, fakeText, fakeText, fakeText, fakeText, fakeText)
names(fake.df.seasons) <- names(seasons)
fake.df.seasons$Ordered <- as.character(fake.df.seasons$Ordered)
fake.df.seasons$Filming <- as.character(fake.df.seasons$Filming)
fake.df.seasons$`First aired` <- as.character(fake.df.seasons$`First aired`)
fake.df.seasons$`Last aired` <- as.character(fake.df.seasons$`Last aired`)
fake.df.seasons$`Novel(s) adapted` <- as.character(fake.df.seasons$`Novel(s) adapted`)

head(fake.df.seasons)
```

```
##   Season Ordered Filming First aired Last aired Novel(s) adapted
## 1      8    fake    fake        fake       fake             fake
## 2      9    fake    fake        fake       fake             fake
## 3     10    fake    fake        fake       fake             fake
## 4     11    fake    fake        fake       fake             fake
## 5     12    fake    fake        fake       fake             fake
## 6     13    fake    fake        fake       fake             fake
```

```
fakeReviews <- c(101:200)
fakeScores1 <- round(runif(length(fakeReviews), min=1, max=100))
fakeScores2 <- round(runif(length(fakeReviews), min=1, max=100))
fake.df.critics <- data.frame(fakeReviews)
names(fake.df.critics) <- c("Season")
fake.df.critics$`Critical response >> Rotten Tomatoes` <- paste0(fakeScores1, "% (33 reviews)")
fake.df.critics$`Critical response >> Metacritic` <- paste0(fakeScores2, " (34 reviews)")

head(fake.df.critics)
```

```
##   Season Critical response >> Rotten Tomatoes
## 1    101                     60% (33 reviews)
## 2    102                     43% (33 reviews)
## 3    103                     70% (33 reviews)
## 4    104                     65% (33 reviews)
## 5    105                     93% (33 reviews)
## 6    106                     34% (33 reviews)
##   Critical response >> Metacritic
## 1                 19 (34 reviews)
## 2                  8 (34 reviews)
## 3                 63 (34 reviews)
## 4                 21 (34 reviews)
## 5                  9 (34 reviews)
## 6                 54 (34 reviews)
```

Ok so what did we do here? We've made a large number of fictious seasons from 8 up to 100, and critics reviews for fictious seasons 101 to 200. Let's add them into the original wikipedia tables we pulled.

```
seasons$Season <- as.numeric(seasons$Season)
criticResponses$Season <- as.numeric(criticResponses$Season)
#we need to relax the factors to be able to include the new "seasons"

seasons <- rbind(seasons, fake.df.seasons)
criticResponses <- rbind(criticResponses, fake.df.critics)

#rebuild factors
seasons$Season <- factor(seasons$Season, levels = c(1:100), labels = c(1:100))
criticResponses$Season <- as.factor(criticResponses$Season)
```

So far so good, but this still a little too convenient, the rows we want, still overlap (they are the first 7), so let's randomly reorder the dataframes.

```
library(dplyr)
seasons <- sample_n(seasons, size = nrow(seasons))
criticResponses <- sample_n(criticResponses, size=nrow(criticResponses))
head(seasons)
```

```
##     Season       Ordered             Filming    First aired    Last aired
## 1        1 March 2, 2010 Second half of 2010 April 17, 2011 June 19, 2011
## 70      70          fake                fake           fake          fake
## 52      52          fake                fake           fake          fake
## 28      28          fake                fake           fake          fake
## 40      40          fake                fake           fake          fake
## 100    100          fake                fake           fake          fake
##      Novel(s) adapted
## 1   A Game of Thrones
## 70               fake
## 52               fake
## 28               fake
## 40               fake
## 100              fake
```

```
head(criticResponses)
```

```
##    Season Critical response >> Rotten Tomatoes
## 81    181       10.1438035077881% (33 reviews)
## 59    159       5.47556190751493% (33 reviews)
## 97    197       64.4534173316788% (33 reviews)
## 40    140       1.66889401781373% (33 reviews)
## 63    163       30.2398245392833% (33 reviews)
## 58    158       46.2293880700599% (33 reviews)
##    Critical response >> Metacritic
## 81   10.5762069681659 (34 reviews)
## 59    25.487805735087 (34 reviews)
## 97   28.7519311457872 (34 reviews)
## 40   92.2892017702106 (34 reviews)
## 63   88.5571205413435 (34 reviews)
## 58   29.1245558995288 (34 reviews)
```

Let's try to merge again and see what we get

```r
GoTWikipedia2 <- merge(seasons, criticResponses, by="Season")
str(GoTWikipedia2)
```

```
## 'data.frame':    7 obs. of  8 variables:
##  $ Season                             : Factor w/ 100 levels "1","2","3","4",..: 1 2 3 4 5 6 7
##  $ Ordered                            : chr  "March 2, 2010" "April 19, 2011" "April 10, 2012" "Apri
##  $ Filming                            : chr  "Second half of 2010" "Second half of 2011" "July - Nov
##  $ First aired                        : chr  "April 17, 2011" "April 1, 2012" "March 31, 2013" "Apri
##  $ Last aired                         : chr  "June 19, 2011" "June 3, 2012" "June 9, 2013" "June 15
##  $ Novel(s) adapted                   : chr  "A Game of Thrones" "A Clash of Kings and some early cl
##  $ Critical response >> Rotten Tomatoes: chr  "89% (33 reviews)" "96% (33 reviews)" "97% (44 reviews
##  $ Critical response >> Metacritic     : chr  "80 (28 reviews)" "90 (26 reviews)" "91 (25 reviews)"
```

All our noise disappeared :) as merge ignores cases where there is no match as specified in the *by* column. I think that's pretty cool. All that's left to do is rebuild the factor, I'll leave you to do that…

Back to GoT proper

Ok so we have our seasons data and also the critics reviews in one data.frame. This was a pretty simple exercise in accessing table-based data from the web. Check this page, if you find you need to do this in the project, it has a few nice examples of less straight-forward situations; https://cran.r-project.org/web/packages/htmltab/vignettes/htmltab.html

I don't know about you, but somehow I don't feel convinced that the wikipedia data is accurate or detailed enough. Let's grab more data from the movie database – it's worth noting that there isn't an R library available (there are python libraries, but this is a good opportunity to explore playing with JSON data). We need an API key for this website, so head over to register and get yourself an API key: https://www.themoviedb.org

```r
library(rjson)
url <- "https://api.themoviedb.org/3/tv/1399?api_key=<put your key here>"
GoTJson <- jsonlite::fromJSON(url, flatten = TRUE)
```

Task

Explore the GoTJson data frame a little and see what we've got – short answer is a data frame of data frames.

So let's grab some details on the seasons.

```r
movieDBSeasons <- GoTJson$seasons
```

Ok, so if we check the website https://www.themoviedb.org/tv/1399-game-of-thrones we see that season 0 is a documentary, so we don't want that.

Task Get rid of season 0.

So we're interested in the *id* column, as that will allow us to further explore the seasons. Let's get some details on season 1:

```r
key <- "<put your key here>"
```

```r
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/1", "?api_key=", key)
GoTJsonSeason1 <- jsonlite::fromJSON(url, flatten = TRUE)
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/2", "?api_key=", key)
GoTJsonSeason2 <- jsonlite::fromJSON(url, flatten = TRUE)
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/3", "?api_key=", key)
GoTJsonSeason3 <- jsonlite::fromJSON(url, flatten = TRUE)
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/4", "?api_key=", key)
GoTJsonSeason4 <- jsonlite::fromJSON(url, flatten = TRUE)
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/5", "?api_key=", key)
GoTJsonSeason5 <- jsonlite::fromJSON(url, flatten = TRUE)
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/6", "?api_key=", key)
GoTJsonSeason6 <- jsonlite::fromJSON(url, flatten = TRUE)
url <- paste0("https://api.themoviedb.org/3/tv/1399/season/7", "?api_key=", key)
GoTJsonSeason7 <- jsonlite::fromJSON(url, flatten = TRUE)
```

Ok so what we have is a data frame of data frames (where each item in our data frame corresponds to one episode) for each season.

We'll start with something "easy" – compute the average rating of each season and compare to the wikipedia data. Then we'll move on to some simple data exploration and visualisation tasks.

Average Review Scores

So let's make an empty vector, and then compute and add in the average review scores for each season

```r
movieDBRatings <- c()
movieDBRatings[1] <- mean(GoTJsonSeason1$episodes$vote_average)
movieDBRatings[2] <- mean(GoTJsonSeason2$episodes$vote_average)
movieDBRatings[3] <- mean(GoTJsonSeason3$episodes$vote_average)
movieDBRatings[4] <- mean(GoTJsonSeason4$episodes$vote_average)
movieDBRatings[5] <- mean(GoTJsonSeason5$episodes$vote_average)
movieDBRatings[6] <- mean(GoTJsonSeason6$episodes$vote_average)
movieDBRatings[7] <- mean(GoTJsonSeason7$episodes$vote_average)

movieDBRatings
```

```
## [1] 7.803200 8.045900 8.004900 8.199100 8.074800 7.905200 8.615143
```

So our wikipedia data is a percentage, so let's make our movieDB score consistent with them.

Tasks

1. Make the movieDBRatings vector a % score (multiply by 10), and round them
2. Add averages into the Wikipedia data.frame *GoTWikipedia* so that we get:

```r
head(GoTWikipedia[, c("Critical.response....Rotten.Tomatoes",
                     "Critical.response....Metacritic", "movieDBRatings")])
```

```
##   Critical.response....Rotten.Tomatoes Critical.response....Metacritic
## 1                        89% (33 reviews)                8O (28 reviews)
## 2                        96% (33 reviews)                9O (26 reviews)
## 3                     97% (44 reviews)                 91 (25 reviews)
## 4                     97% (50 reviews)                 94 (29 reviews)
## 5                     95% (52 reviews)                 91 (29 reviews)
## 6                     96% (29 reviews)                  73 (9 reviews)
##   movieDBRatings
## 1             78
## 2             80
## 3             80
## 4             82
## 5             81
## 6             79
```

Let's fix the wikipedia scores so that we have numerical data as opposed to strings, then we can plot the data and compare. Let's be pretty pragmatic here, we actually only need the first 2 characters of the wikipedia data:

Task Make two new attributes *RT* and *Critic* that are the first 2 characters of the corresponding *GoTWikipedia* data.frame and cast them to numerics as shown below.

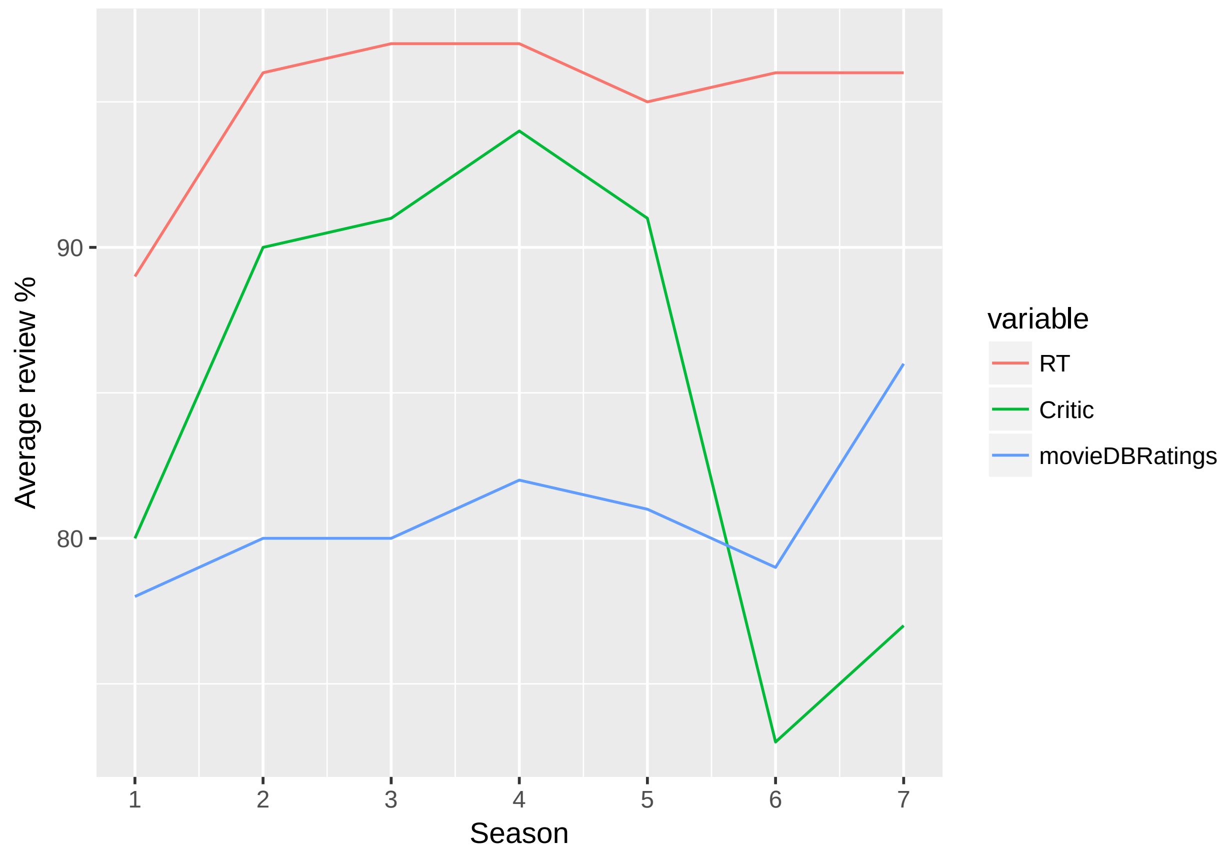
HINT use the substr function

```r
head(GoTWikipedia[, c("RT", "Critic", "movieDBRatings")])
```

```
##   RT Critic movieDBRatings
## 1 89     80             78
## 2 96     90             80
## 3 97     91             80
## 4 97     94             82
## 5 95     91             81
## 6 96     73             79
```

Now let's plot them against each other

```r
library(ggplot2)
library(ggthemes)
library(reshape)
data <- data.frame(GoTWikipedia[, c("Season", "RT", "Critic", "movieDBRatings")])
data$Season <- as.numeric(data$Season)
Molten <- melt(data, id.vars = "Season")
ggplot(Molten, aes(x = Season, y = value, colour = variable)) + geom_line() +
  scale_x_continuous(breaks=c(1:7)) + ylab("Average review %")
```



OK so what did we actually do here?

1. Pull two tables from Wikipedia that captured aggregate reviewer scores on the 7 HBO GoT Seasons
2. Pull a JSON object representing each series from https://www.themoviedb.org/tv/1399-game-of-thrones
3. Clean up the data a little
4. Compute aggregate reviews from the JSON objects for each GoT Season
5. Merge the Wikipedia data and the themoviedb data
6. Melt the merged data into a lower dimensionality (view the *Molten*) object to see what this means
7. Convert the Seaon factor back to a numeric such that it makes sense as a value on the x-axis
8. Plot all 3 reviewer scores (from 2 different data sources) against each other

Amd all this without once going File -> Save As :)

Explore Deeper into the JSON objects

Ok so we have some fun scratching the surface of our themoviedb data, now let's go deeper into the JSON object and see what we can find. To guide us, let's start with a simple question. Are there more male, or female extras/guest stars in GoT?

Let's go back to our JSON object:

```r
str(GoTJsonSeason1)
```

Inside the nested episodes dataframe we have some nested dataframes called crew, and there is a gender attibute hidden in there too, so all we need to do is traverse across all episode within all seasons and we'll have our answer.

```r
s1 <- GoTJsonSeason1$episodes$guest_stars
```

This object is a list of data.frames (see: str(s1)). So we are going to loop through it. We could do this with lapply or a for loop, as we haven't done for loops yet, let's use one of those.

```r
s1gender <- c()
for (i in 1:length(s1)) {
  tmp <- s1[[i]]
  s1gender <- c(s1gender, tmp$gender)
}
```

We don't want to run this 6 times, so let's introduce the idea of making a function.

```r
getGender <- function(seasonJSON) {
  gender <- c()
for (i in 1:length(seasonJSON)) {
  tmp <- seasonJSON[[i]]
  gender <- c(gender, tmp$gender)
}

return(gender)
}
```

Let's call our function now for season 2

```r
s2gender <- getGender(GoTJsonSeason2$episodes$guest_stars)
```

Tasks

1. Repeat for the other 5 seasons making a factor vector s3gender, . . . , s7gender
2. Compute the number of each gender and instantiate attibutes in our main GoT data.frame corresponding to 0 (unknown), 1 (female), 2 (male)
3. Plot Gender against Season to work out how the gender of extras / guest stars changes over the seasons

---

Exercises

1. Plot gender distributions by season
2. Is there a relationship between the number of (fe)male extras and the Season ratings?
3. Do more extras tend to result in better ratings?
4. What are the most common words in season / episode overviews?
5. Which season has the most positively phrased overview?