GIE Final Project: Flight Scrapping

Group 2 - Carles Mitjans, Ernest Benedito

7 de junio de 2017

Index

1 Introducion and goals

There are webpages such as Skyscanner that scraps different flight sites in order to tell you the cheaper flights between two cities. Our goal is not to replicate it, but to improve a feature that the web doesn't have.

The thing is that with the web you can only look for the flights in one date and one location at the time. So if, for instance, you have flexibility of dates to make your trip (imagine you can depart in a range of 3 days and come back in another range of 3 days) and you do not have any preferences between going to either some different airports inside the country you are visiting (imagine there are 5 airports which you are fine to go because you are doing a trip around the country), you have to search date by date and airport by airport in order to see all possibilities (for our example would be 45 different searches).

Our scope is that, given the range of days you are able to depart, the range of the days you are able to come back, the city of departure and the country you are visiting, our algorithm ranks you the flights sorted by its price.

2 Data extraction

Initially, we planned to use the Skyscanner page itself, but at the moment of scrapping, it resulted that they don't allow to webscrap from their webpage, at least not easily, as they protect their data and are able tu detect when an extraction is being done through webscrapping.

Thus, we decided to work with *edreams*, which offer a very similar survey and are opened to webscrapping. One of the first issues to handle is that when a search is done in the web, as it works as an API there is a significant lad time, so it is not possible to extract the information on the first instance, but you have to wait until the page is fully loaded. Thus, an additional feature had to be handled, using a JavaScript structure through *phantomjs*, which allows to access to a web page and wait some time to load.

Thus, combining the JavaScript functionallity along with the R package *rvest*, we were able to extract an *html* file with the webpage once the query is loaded, and with this *html* we could extract cheapest price of a flight given it's departure and arrival airports and dates.

The function that does this is the following:

```
require(rvest)

cheapest <- function(dfrom, dto, departure, return){
  ## change Phantom.js scrape file
  url <- paste0('http://www.edreams.es/#/results/type=R;dep=', departure, ';from=',
                dfrom,';to=', dto, ';ret=', return,
                ';collectionmethod=false;airlinescodes=false;internalSearch=true')
  lines <- readLines("scrape_final.js")
  lines[1] <- paste0("var url ='", url ,"';")
  # Here we set the waiting time to 40,000 miliseconds (40 seconds)
  lines[14] <- paste0('    }, 40000);')
  writeLines(lines, "scrape_final.js")

  ## Download website
  system("phantomjs scrape_final.js")

  ### use Rvest to scrape the downloaded website.
  pg <- read_html("1.html")

  # Get the nodes with the prices of the flights for the given combination
  prices <- html_attr(html_nodes(pg, ".od-pricebreakdown-results-passengers-row"),
```

```
                      "data-item-price")

  prices <- as.double(prices)

  # Get the minimum price
  price <- min(prices)

  return(price)
}
```

Remember that this function has as inputs the airports and dates of departure and return and the output is the price of the cheapest flight for that combination. We have to notice as well that we set the waiting time to load the page to 40s, this slows the whole process but is a time to make sure that with a bad connection you get everything loaded. If we wanted to industrialize this product we should shorten this time.

3 Creation of a comparison funciton

Once we were able to extract the criteria we wante for our sorting, it was time to create the desired function. In order to recapitulate, we must remember that our function has the inputs:

- Airport of departure
- Country of arrival
- Earliest departure possible date
- Latest departure possible date
- Earliest returning possible date
- Latest returning possible date

And would return a dataframe with the following variables:

- From: Airport of departure
- To: Airport of destination
- Municipality: Municipality of the destination airport
- Depart: Date of departure
- Return: Date of return
- Price: Cost of the ticket ($)

3.1 Getting destination airports

The first step is that, given a country, you get in return the possible airports of that country. To do so, we looked for a database with all registered airports, which has got, among other variables, the *ISO Country*, which contains the country, and the *IATA Code* of the airport, which all airports have a unique one.

We thus load this data and clear it:

```
airports <- read.csv('airport-codes.csv')

# Delete missing IATA
airports <- airports[!(airports$iata_code %in% c("", "-", "---", "0")),]

# Keep important variables
airports <- airports[,c(3, 8, 10, 12)]

# Delete duplicated rows
n_occur <- data.frame(table(airports$iata_code))
```

```r
repeated <- n_occur[n_occur$Freq > 1,1]
airports <- airports[!(airports$iata_code %in% repeated),]
airports <- airports[complete.cases(airports),]
head(airports)
```

```
##                         name iso_country  municipality iata_code
## 210          Utirik Airport          MH Utirik Island       UTK
## 641  Buckhorn Ranch Airport          US Crested Butte       CSE
## 1050      LBJ Ranch Airport          US  Johnson City       JCY
## 1350   Metropolitan Airport          US        Palmer       PMX
## 1386   Loring Seaplane Base          US        Loring       WLR
## 1501    Nunapitchuk Airport          US   Nunapitchuk       NUP
```

3.2 Implementation

Thus, the function that combines everything is the following:

```
comparator <- function(IATA_from, ISO_to, dep1, dep2, ret1, ret2){
  # Get name of departure airport
  from <- airports[airports$iata_code == IATA_from,]$name[1]
  # Get possible arrival airports given the country with its ISO code
  possible_airports <- airports[airports$iso_country == ISO_to,]
  output_dataframe <- data.frame(From = character(),
                                 To = character(),
                                 Municipality = character(),
                                 Depart = character(),
                                 Return = character(),
                                 Price = double())
  possible_depart <- seq(as.Date(dep1), as.Date(dep2), by = 'days')
  possible_return <- seq(as.Date(ret1), as.Date(ret2), by = 'days')
  for (depart in possible_depart){
    for (return in possible_return){
      for (i in 1:dim(possible_airports)[1]){

        # Looks the cheapest price for that combination
        price <- cheapest(dfrom = IATA_from, dto = as.character(possible_airports$iata_code[i]),
                          departure = as.character(depart), return = as.character(return))

        # Adds new row to output dataframe
        output_dataframe[nrow(output_dataframe) + 1,] = c(from,
                               possible_airports$name[i],
                               possible_airports$municipality[i], depart,
                               return, price)
      }
    }
  }

  # Sorts by price
  output_dataframe[with(output_dataframe, order(Price)), ]
  return(output_dataframe)
}
```

3.3 Example

With all this, let's make an example in order to see if it works well. First we are going to check if the *cheapest* function works well. This function returns the cheapest flight between two airports for a given dates. We'll asume we are flying from Barcelona to Pisa the 1st of August and return the 8th.

```
cheapest('BCN','PSA', '2017-08-01', '2017-08-08')
```

```
## [1] 110.98
```

As we can see, the cheapest flight for that specific combination is 110.98 ??? according to the *eDreams* search. As stated before, this lasted around 40 seconds to compute as that's the waiting time we set in the beggining. For this example it is not too much, but for the next one the full search can take several minutes.

Secondly, let's check if the *comparator* also works well. For our case, we'll assume we are departing from Barcelona - El Prat, with IATA code *BCN*, and want to go to any airport in the Czech Republic, with ISO

code *CZ*.

We are flexible to depart from 1st to 3rd of July and are able to return from 14th to 15th of July, for instance, then we'd do:

```
df <- comparator(IATA_from = 'BCN', ISO_to = 'CZ', dep1 = '2017/07/01',
                 dep2 = '2017/07/03', ret1 = '2017/07/14', ret2 = '2017/07/15')

head(df, 10)
```

```
##    From   To Municipality Depart Return Price
## 1  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 2  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 3  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 4  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 5  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 6  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 7  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 8  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 9  <NA> <NA>         <NA>   <NA>   <NA>   Inf
## 10 <NA> <NA>         <NA>   <NA>   <NA>   Inf
```

Thus, we can see that it worked, sorting the flights for all combinations by their price. In the previous table we see, then, the ten cheapest flights and their respectives dates and destination airports for the given search we made.

4 Conclusions

With this study we have been able to combine already existing data with html and web scraping technologies in order to obtain a combination of possible cheap flights between two destinations and dates. This feature isn't available in flights-booking web services as the ones used in this study, which gives an additional value to its usability.

The biggest issues found during the development of this project are the time used by the program to get the results and the use of Javascript methods by the webpage. Thet time used, though, could be hugely improved if we decreased the waiting loading time of the webpage, which was set to 40 seconds for modalization purposes, but could be much less.

All in all, we successfully achieved our goals and were able to design a service which could be useful in the world of flight searches engine.