Other applications 4: accessing websites part II

Let us now assume you would like to compile a web-based corpus on a particular topic, let's say on the programming language Perl. Let us further assume that you do not want to look around for suitable pages on your own but rather use Google (or some other search engine) to give you appropriate pages on this topic.

Assignment 1

Write a script that has the following characteristics and performs the following operations:

(i) The script defines a search word ("perl") and a number of hits (30) to be retrieved from a search engine (such as Google).

(ii) The script pastes this together into a URL to be sent to Google. With Google, the URL begins with "http://www.google.com/search?q=" followed by the search word, followed by "&num=" followed by the number of desired hits.

(iii) The script sends this URL to Google and loads the search results into a character vector `search.site`. It then retrieves the lines with the main recommendations, i.e., not the links to filters, cached pages etc. The main recommendations are hyperlinks that begin with "<a class="l href=".

(iv) The script then outputs a vector with all the links. This vector could then serve as input for another script or for additional lines to actually load these files with scan as in the first assignment of this section. (This would correspond to the functionality of the WebGetter utility in WordSmith Tools 4. Of course, such a 'corpus' would still have to be cleaned from HTML tags, Java/CGI scripts etc., but of course this is easy with R, too.)

When you are done, load `<C:/_qclwr/_scripts/other_4_assignment1_links_google.r>` and compare your solution with them.

The above approach is a little limited: whatever number of hits you specify, Google maximally returns 100 links. However, we may want to get up to 1,000 links, which is the maximum number of hits Google returns and which is what the following assignment is all about.

Assignment 2

Write a script that has the following characteristics and performs the following operations:

(i) The script defines a search word ("perl") and a number of hits to be retrieved from a search engine (such as Google).

(ii) The script determines how many times it has to paste together a URL and send it to Google to recover the desired number of links.

Note: depending on how you approach this task, you may have to cope with desired numbers of links that are multiples of 100 (e.g., 500) or not (e.g., 550). If this issue does arise and you find it too difficult, just disregard it for now.

(iii) In a loop, the script pastes together all URLs to be sent to Google. This time, try to find out yourself what the URLs to be sent to Google to get links 101 to 200, 201, to 300, etc.
must look like (by doing a Google search in your browser and seeing what happens if you request the next 10 hits).

(iv) The script successively sends these URLs to Google, loads the search results into a character vector, and retrieves the lines with the main recommendations, i.e., not the links to filters, cached pages etc. The main recommendations are hyperlinks that begin with "<a class=l href=".

(v) The script collects all the hyperlinks in one vector and outputs that vector, which again could serve as input for another script or for additional lines to actually load these files with \texttt{scan}. If you want to play around with this a little more, why don't you enter your name as a search word and add an additional line to the script which tells you how many hits come from each domain name?

When you are done, load \texttt{C:/qclwr/scripts/other_4_assignment2_links_google.r} and compare your solution with them.