How to Automate the Ancestry Group Tagging Using a Web Browser

This little guide refers to the version 2 of the autotagger script. This version of the script has more error checking and should hopefully run a little better. I advice you to use this version. For reference I link the original script:
https://drive.google.com/file/d/12VNTvx93d1biNj23mPz6txK3vbfLadjz/view?usp=sharing

Document Version History

- 0.1 - 30 Jul 2020 - Initial version based on autotagger script v2

Overview

This is an example of a possible way to automate the group tagging of DNA matches on Ancestry. It uses a small piece of Javascript and takes advantage of the infinite list functionality on the page to keep getting new matches to tag.

This is somewhat of a hack that uses the browser devtools console to interact with the DNA matches page rendered in your browser. There are probably other (perhaps better) ways to do it but this one is simple and requires no additional software/tools – just your browser and a simple text editor like Notepad in Windows or similar. It will work on a Mac as well as in Windows (or Linux).

Although it may seem overly complicated it is actually not that bad. You basically only have to provide two pieces of information to the script and then do some copy paste to get it to work.

I have used this to tag my two kits. Running for hours and hours and with a few restarts here and there it auto-tagged over 50000 matches for me.

You can do this too!

Caveat

The script may crash out of the blue. It can be due to network hiccups, the old trusty “backend overtaxed” message from the server and countless other reasons. It can also just tax the browser tab memory itself so hard it eventually crashes. When this happens just follow the method described to restart and continue.

As far as browser to use goes you will get by with most modern ones. Chrome, Firefox and the new Chromium-based Edge browser will all work. Not sure about Safari. A 64-bit version

is preferable since it will have a larger amount of memory allocated to it's just a nice-to-have so don't worry.

A very old browser that hasn't been updated in ages may give you problems. Just saying.

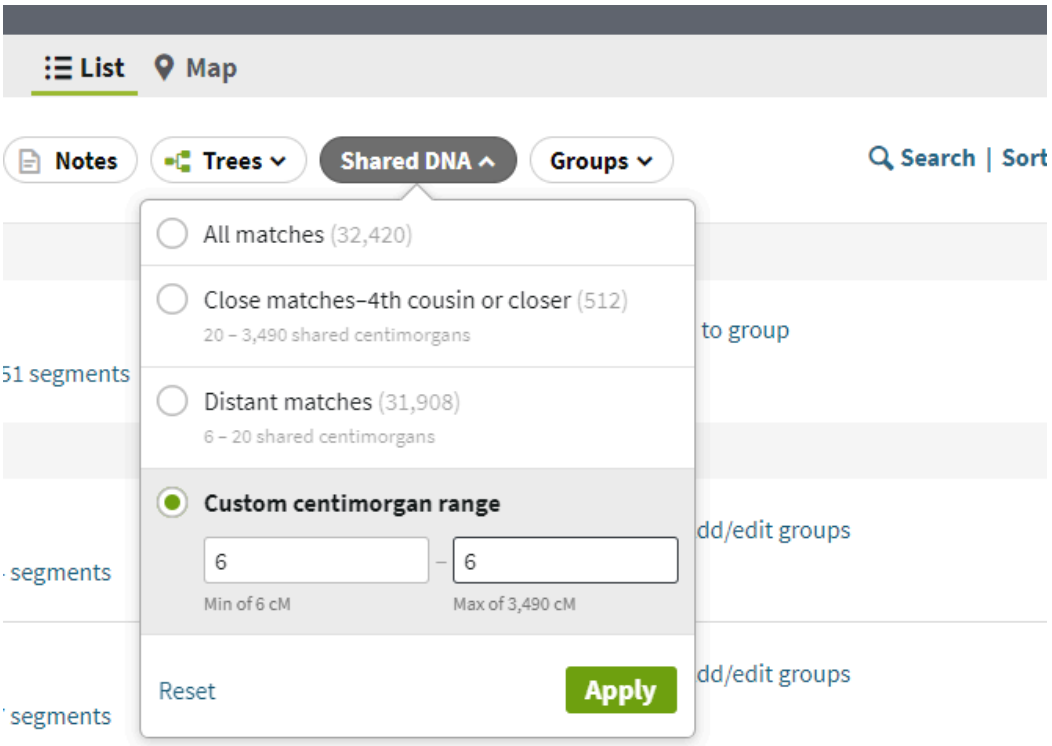
Please read through the whole thing to get an overview.

Prepping Steps

There are a few small things that must be done by you to get going.

This is a step-by-step list of how to prepare for running the script:

- First log into Ancestry.com using your browser.
- Open the DNA matches page and filter the matches to your liking (ill.1). Note that your filter should provide as few matches in total as possible. I.e. set the filter to 6-6cM for instance and not 6-8cM. Do 6, then 7, then 8 – not all at the same time. This will strain your browser tab's memory too much. The page will load the initial few matches which is all the script needs to get going on its own.



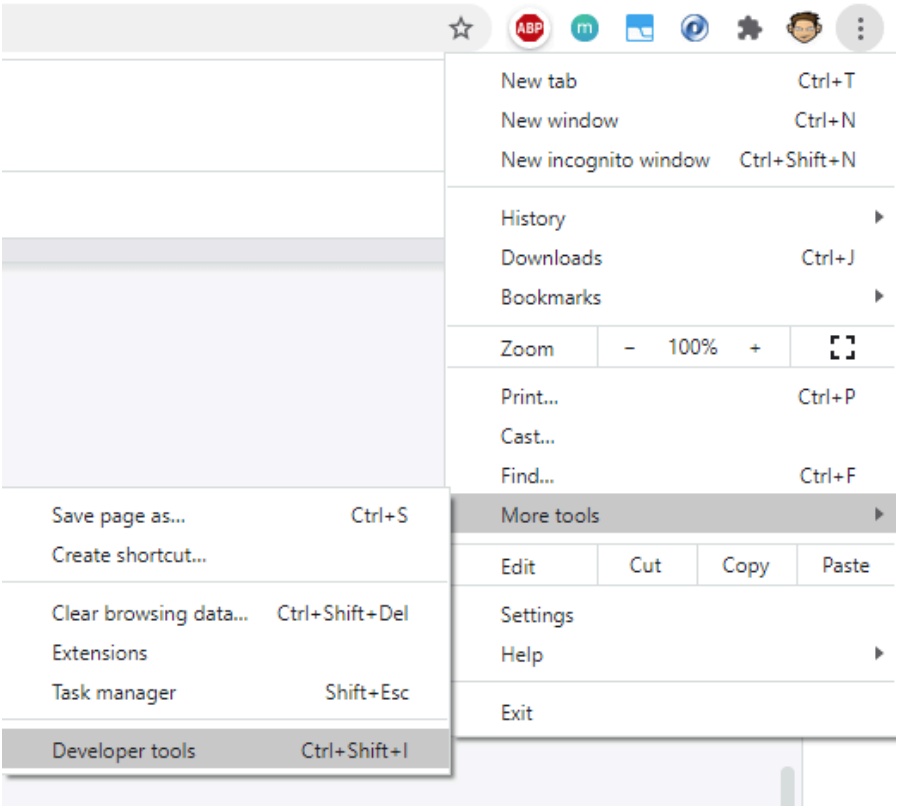
Ill. 1: Filter your matches

- Now you need to open the developer console in the browser. There are several ways to do this:

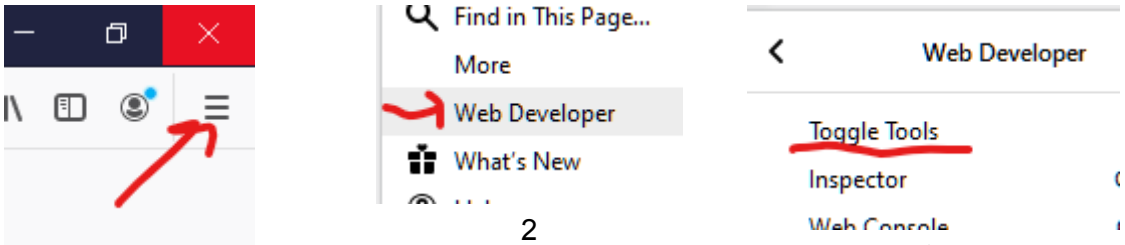
- Clicking the F12 key on your keyboard (works in Chrome and Firefox) but may not work if your laptop if set up to with media keys. There usually is a “function key” (FN) that you can hold down.

- Using the CTRL+SHIFT+I (that’s an i) key combination (works in Chrome and Firefox)

- Or clicking your way in the menus (ill. 2 and 3)



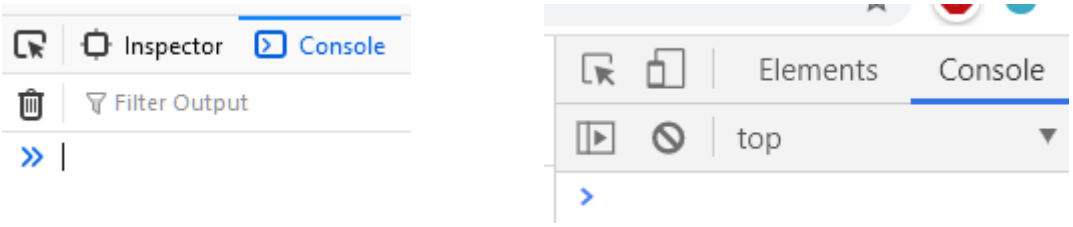
Ill. 2: Opening the developer tools in Chrome



1 2 3

Ill. 3: Opening the developer tools in Firefox in 3 clicks

- In this devtools panel locate and press the «Console» tab (ill. 4). This is a text-based command-response dialog. Commands are written in Javascript which is the "magic" that makes web pages interactive (more or less).



Ill. 4: Console in Firefox (left) and Chrome (right)

- The **>** and **>>** marks the input line. You click on this line to put focus in the console to write commands or paste larger script texts (which we will do shortly).

  As an example you can run your first script command in the console like this:

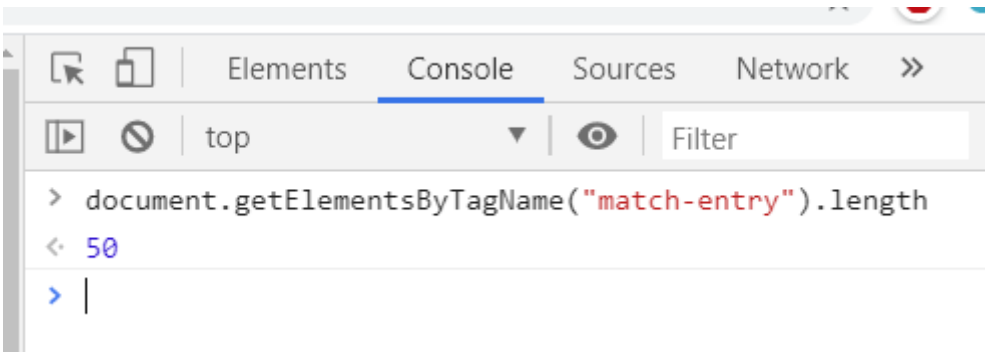
  - First mark the below script line and copy it

    ```
    document.getElementsByTagName("match-entry").length
    ```

  - Then click on the console input line next to the ">>" line to set focus (you will see a blinking black cursor when the console has focus)

  - **A note for Firefox users**: Firefox may not allow you to paste into the console by default. If this happens then you must first manually write into the console the two words: **allow pasting**. After that you can paste anything you like.

  - Then paste the script line and press Enter. This will output the number of DNA matches currently loaded in your browser tab - 50 for an initial matches list (ill. 6). Pretty easy, huh? Now you're ready to tag some matches.



Ill. 5: Running a test command in the console

Method

The general method for applying the script:

- Make sure the script has been adjusted for your setup according to chapter «Preparing the Script for Your Setup» below

- Copy the prepared script from the text editor and paste it into the console. Then simply press Enter to execute it. Make sure the console has focus by clicking the «>» line first

- The script will works its way down the list - tagging one and one match, and since it's an infinite autoloading list the page itself will fetch more matches as the script gets close to the page bottom thus allowing the script to keep working uninterrupted.

- The script MAY (will) fail. When this happens you may be able to restart it again by putting focus in the console and pressing the "arrow up" key. This will bring up the previous command (the script) and you just press Enter again to start. However, the browser tab may also have "died". If that's the case you have to open a new tab and apply the script to that tab instead. Run the same set of steps in the new tab.
  The script will start to auto-scroll and load new matches until it finds some untagged matches and continue tagging.

- The script will finish when it has tagged all the matches in the list. This happens either due to some "backend overtaxed" situation where Ancestry doesn't send a new set of matches or when the script actually reaches the bottom of the list (and scrolling does not provide any more matches).

- If you want to run the script on another set of matches you should close the tab and load the new match set in a new tab. This is to start fresh with "new" tab memory to work with. Then reapply the procedure again in the same way.

A Few Tips

- If you're having problems have a look through the troubleshooting chapter at the end
- In the Windows settings you can adjust the power settings. You should temporarily set these so that your machine doesn't shut down.
- In Chrome you can adjust the settings for the site and disable loading of images. Go to Settings->Privacy & Security->Site Settings. Scroll down and click into the Images section. There you can toggle the "Show all" to "Do not show any images". This will prevent the page from loading the profile photos. This reduces the amount of data fetched from the server.

About Running Multiple Instances

Yes, you can. Either multiple lists for the same match. For instance one list with 6 cM matches and one list with 7s, etc. Or perhaps several different kits. It mostly comes down to the capacity of your computer and your internet connection speed.

Preparing the Script for Your Setup

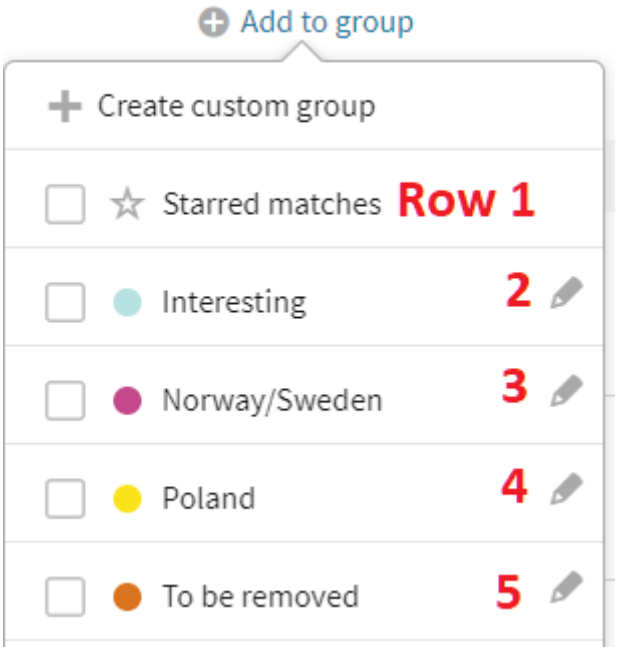
Since you and I probably use different group name and have a different ordering of the custom groups you need to adjust the script slightly. It's as simple as two small changes.

- Start Notepad or some other simple text editor. See the subchapter for a word of caution about "smart" text editors.
- Copy the auto-tagger script from chapter «Auto-Tagger Script» into your text editor
- You need to change the first two lines of the script to match your group settings.

```
// MODIFY THE FOLLOWING LINES AS NEEDED
var groupTitle = "To be removed";
var groupRow = 2;
```

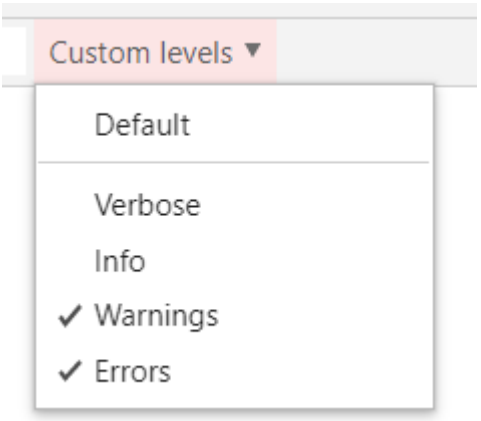
Ill. 6: Adjusting group settings in the script text

  - Change the group title to your own group name that you will use. Make sure to only change the text inside the quotation marks. Make it identical with respect to small/capital letters. Use a simple name - preferably only letter and numbers
  - On the groupRow line replace the number with a number that is the row number of **your** custom group (in the selector popup that displays when you click «Add/edit groups» - see ill. 8). The star is 1, first custom row is 2, etc. Count your way down to the group you will use.



Ill. 7: How the script sees the rows in the group selector

- You can also enable/disable logging of progress information into the console during script execution with the doMatchLogging flag. Change the word “true” to “false” to disable and vice versa.
  If you let logging be enabled you should also adjust the log level in the console so that you primarily see output from the script. You can do this by unchecking/unmarking the verbose log levels and only leave Warnings and Errors enabled. The script writes log lines with a Warning level so that we don’t have to get spammed with the Ancestry verbose logging



Ill. 8: Console log levels in Chrome. Only Warnings/Errors enabled.

Note that the script text formatting doesn’t matter as long as:

- No characters are replaced by similar looking ones
- No script keywords are split in any way - for instance by an accidental line break

- All the semicolons are in place. They are used by Javascript to separate instructions.

A Note About Editors

Some text editors try to be “helpful”. They may do stuff like replacing a character to make it look prettier. That is absolutely not helpful in a programming language. The text of the script cannot be replaced by similar looking characters. The Javascript engine in the browser will not recognise the commands and the script will fail.

The most commonly seen error is that when editing the group title line in the script it suddenly gets a different type of quotation mark. Keep an eye on this.

Auto-Tagger Script

I put the script in a separate text file since it fits poorly into this document style. Note that this is a link to a code repository that I push changes to. Fixes and updates will be added to this file continuously.

Access it via this link:

https://github.com/lrf1/ancestry_scripts/blob/master/ancestry_dnsmatches_grouptagger_v2.js

Copy EVERY LINE of the script into your text editor. Missing just one character in the beginning or end will break the script execution.

A little tip is to click the little “Raw” button on the page. That will give the script text the focus of the whole page and make it easier to copy.

Typical Problems Seen

- The script may just crash. Try to restart it a few times. Perhaps open a new tab to run it in.

- Like mentioned previously some editors may replace quotation marks with a type that is non-compatible with the browser’s script engine. Double-check the changes made when editing the script.

- Some people have reported that the group row used didn’t correspond with the star being line 1. If this is the case then just adjust the row number as you need to hit the group you want.
  If it starts using the wrong group just reload the page to stop the script. Then adjust

and paste it again.

- Opening the console window is described elsewhere.

- Some people seem to have had issues getting the script going using Firefox. You can always download Chrome and try that or the new Windows Edge Chromium browser which is basically a Chrome with a little Windows sprinkled on top.

- Some people have reported that the script runs a little and stops. It doesn't autoscroll as expected. If you can't solve this problem then you can bypass it by preloading the entire list before running the autotagger script.

  Paste the below script snippet into the console and press Enter to scroll for the specified number of repetitions. Note that once started the script can't be stopped without reloading the page so it's better to run it a few times with fewer repetitions than once with way too many.

  Once the page has reached the end of the list the script will still apply the remaining number of repetitions trying to scroll further before it finishes.

```
(async() => {
var numTimes = 100;
var waitTimeInMS = 3000;
for (let i = 0; i < numTimes; i++) {
window.scrollTo(0, window.document.body.scrollHeight);
await new Promise(r => setTimeout(r, waitTimeInMS));
}
}) ()
```