Table of Contents

Ancestry parser 2

Ancestry parser

```
(async function (minSharedDna, maxSharedDna) {
    const accountID = window.location.href.replace(/\?.*/,
"").split(/\//).pop();
    let initPage = 1;
    let skipList = ["Name One", "Name Two"];
    let matchCounter = 0;
    var data = "";

    if (!accountID) {
        console.error("Running script on wrong page!");
        return 1;
    }

    function saveMatches() {
        console.log("Saving data to file...");
        if (typeof data === "object") {
            data = JSON.stringify(data, undefined, 4);
        }
        var blob = new Blob([data], {type: 'text/csv;charset=UTF-8'});
        e = document.createEvent('MouseEvents');
        a = document.createElement('a');
        a.download = "dnamatches.csv";
        a.href = window.URL.createObjectURL(blob);
        a.dataset.downloadurl = ['text/csv', a.download, a.href].join(':');
        e.initMouseEvent('click',true,false,window,
0,0,0,0,0,false,false,false,false,0,null);
        a.dispatchEvent(e);
    }

    function handleErrors(response) {
        if (!response.ok) {
            throw Error(response.status);
        }
        return response;
    }

    async function processCommonMatches(match, page) {
        if (!skipList.includes(match.publicDisplayName)) {

            // Get common matches for this match
            await fetch(`/discoveryui-
matchesservice/api/samples/${accountID}/matches/list?page=${page}&relationgu
id=${match.testGuid}&sortby=RELATIONSHIP&_t=${Date.now()}`
            )
            .then (handleErrors)
            .then((resp) => resp.json())
            .then((json) => {
            matchCounter++;
```

```
            const commonMatches = [];
            for (const group of json.matchGroups) {
                for (const cmatch of group.matches) {
                    commonMatches.push(cmatch);
                }
            }
            let matchString = match.publicDisplayName;
            if
((match.publicDisplayName).localeCompare(match.adminDisplayName) !== 0) {
                matchString += "(Managed by " + match.adminDisplayName +
")";
            }
            console.log(matchString + "[" +
match.relationship.sharedCentimorgans + "cM] has " + commonMatches.length +
" matches.");
            for (const cmatch of commonMatches) {
                if (!skipList.includes(cmatch.publicDisplayName)) {
                    let commonMatchString = cmatch.publicDisplayName;
                    if
((cmatch.publicDisplayName).localeCompare(cmatch.adminDisplayName) !== 0) {
                        commonMatchString += "(Managed by " +
cmatch.adminDisplayName + ")";
                    }
                    let matchline = matchString +"," + commonMatchString +
"\n";
                    data += matchline;
                }
            }
            return true;
            })
            .catch((code) => {
                console.log("Got error " + code + "when fetching common
matches for " + match.publicDisplayName + "(match # " + matchCounter + ").
Retrying in a bit.");
                matchCounter--;
                return false;
            });
        } else {
            // Just silently skip this match
            return true;
        }
    }

    async function loadPage(page, bookmarkData) {
        while (!bookmarkData || bookmarkData.moreMatchesAvailable) {
            await fetch(`/discoveryui-
matchesservice/api/samples/${accountID}/matches/list?page=${page}&minsharedd
na=${minSharedDna}&maxshareddna=${maxSharedDna}&sortby=RELATIONSHIP&_t=${Dat
e.now()}${bookmarkData?
`&bookmarkdata={%22moreMatchesAvailable%22:true,%22lastMatchesServicePageIdx
%22:${bookmarkData.lastMatchesServicePageIdx}}`: ""}`
```

```
            )
            .then (handleErrors)
            .then((resp) => resp.json())
            .then(async (json) => {

                console.log("Got page #" + page);
                const matches = [];
                for (const group of json.matchGroups) {
                    for (const match of group.matches) {
                        matches.push(match);
                    }
                }
                console.log(`Added ${matches.length} matches from page
${page} request`);

                while (matches.length) {
                    const match = matches.shift(); // Pop match from array
front
                    let processStatus = await processCommonMatches(match,
page);
                    if (processStatus === false) {
                        console.warn("Failed to handle match " +
match.publicDisplayName + "- retrying...");
                        matches.unshift(match); // Just redo this match?Does
this work?
                        await new Promise(r => setTimeout(r,
Math.floor(Math.random() * 10000) + 10000));
                    }
                }
                // Worked through this page set - now set up for new
iteration
                page += 4;
                bookmarkData = json.bookmarkData;
            })
            .catch(async (code) => {
                console.error("Page load error " + code + ". Retrying...");
                await new Promise(r => setTimeout(r,
Math.floor(Math.random() * 10000) + 10000));
            });
        } // end of while
    }

    await loadPage(initPage, null);
    console.log("DONE! Processed " + matchCounter + " matches between " +
minSharedDna + " and " + maxSharedDna + " cM.");
    saveMatches();

})(30, 3490); // min cM, max cM
```