Lab 2: Data Parsing and Manipulation; Git/GitHub

You can't make visualizations without data, and a visualization is worthless if other people can't see and use it. In this lab we're going to cover the basics of data parsing and manipulation, and put it all on GitHub.

You may work together on this project, but your code must be your own and you must submit individually.

The goal is to download the UCI Zoo Dataset, write code to convert the data into a human/machine-readable CSV format, and post the resulting code, input, and output on a GitHub repository.

The problem is that the UCI Zoo Dataset looks like this:

```
aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,1
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,1
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,4
```

This is certainly machine-readable, but it's difficult for a human to tell what's going on. Of course there are advantages to storing data in compact machine-readable formats when the data is large, but this data is *not large at all*.

The goal then is to write code (likely Processing, but not necessarily) to convert it to a readable CSV. Specifically, the output should look like:

```
animalname,hair,feathers,eggs,milk,airborne,aquatic,predator,toothed,...
aardvark,true,false,false,true,false,false,true,true,...
```

To accomplish this, you will need the zoo.data and zoo.names files.

Notice that the `zoo.names` file is relatively unstructured. You will need the Attribute info in item number 7 of this file. Store these attribute names in an array. You will also need the types of each attribute (Boolean, Numeric, etc.). Store those in a separate array.

Next, you will need the `zoo.data` file itself. In your conversion program, load the data (use loadTable if you're using Processing). After the data is loaded, loop through the rows and convert each data value using the "types" array: Boolean becomes `true` or `false`, Numeric stays numeric. (Note that the loadTable example code demonstrates how to loop through rows). Afterwards, print the attribute names array in a comma-separated format, followed by the new dataset (also comma-separated).

You can write your conversion program in anything: Processing, Node.js, Python, R, Java, C, perl, Go, Lisp, Lua, Erlang, and so on.

Once that's done, you need to post your code, the input file, and the output file on GitHub.

(Note: If you're unfamiliar with Git and GitHub, see the UBC STAT 545 Topics and Git pages. Students can sign up to get a free Micro account. If you get stuck, grab one of the Git experts in class.)

Create a new repository named "BioVis-Data-Lab". Clone the repository to your machine and copy your files to the repository folder. Add and commit the files, then push them to GitHub. You'll need to submit a link to your repository, e.g. `https://github.com/YOURUSERNAME/BioVis-Data-Lab/`.

Extra credit: I will give up to 5 extra points if you *send* pull requests and file issues on other students' repositories in the class. I'll give 1 point per pull request / issue, but you have to mix it up– there is a limit of 3 issues and/or 3 pull requests. To get credit, add a section to your readme that looks like:

```
Extra Credit
---

type,link
issue, <link>
pull, <link>
...
```

Requirements

1. Conversion code for the Zoo Dataset.
2. A repo containing your code, input files, and output files. This should be in a "ready to run" state so that I can reproduce the output file.
3. Add a "readme.md" file to your repo with instructions on how to run your converter. (Note: "md" stands for Markdown.)

Turning in the project

Submit link to your repo on myWPI.

Grading

This lab is graded on a 100 point scale. The repo will be graded based on completeness (100).

Links

- GitHub: How to write the perfect pull request