2014-09-20

Counting and mining research data with Unix

By James Baker and Ian Milligan

Reviewed by Melodee Beals, Allison Hegel, and Adam Crymble
Recommended for Intermediate Users

Counting and mining research data with Unix

Introduction

This lesson will look at how research data, when organised in a clear and predictable manner, can be counted and mined using the Unix shell. The lesson builds on the lessons "Preserving Your Research Data: Documenting and Structuring Data" and "Introduction to the Bash Command Line". Depending on your confidence with the Unix shell, it can also be used as a standalone lesson or refresher.

Having accumulated research data for one project, a historian might ask different questions of that same data when returning to it during a subsequent project. If this data is spread across multiple files - a series of tabulated data, a set of transcribed text, a collection of images - it can be counted and mined using simple Unix commands.

The Unix shell gives you access to a range of powerful commands that can transform how you count and mine research data. This lesson will introduce you to a series of commands that use counting and mining of tabulated data, though they only scratch the surface of what the Unix shell can do. By learning just a few simple commands you will be able to undertake tasks that are impossible in Libre Office Calc, Microsoft Excel, or other similar spreadsheet programs. These commands can be easily extended for use with non-tabulated data.

This lesson will also demonstrate that the options for manipulating, counting and mining data available to you will often depend on the amount of metadata, or descriptive text, contained in the filenames of the data you are using as much as the range of Unix commands you have learnt to use. Thus, even if it is not a prerequisite of working with the Unix shell, taking the time to structure your research data and filenaming conventions in a consistent and predictable manner is certainly a significant step towards getting the most out of Unix commands and being able to count and mine your research data. For the value of taking the time to make your data consistent and predictable beyond matters of preservation, see "Preserving Your Research Data: Documenting and Structuring Data".

Software and setup

Windows users will need to install Git Bash. This can be installed by downloading the most recent installer at the git for windows webpage. Instructions for installation are available at Open Hatch.

OS X and Linux users will need to use their terminal shells to follow this lesson, as discussed in "Introduction to the Bash Command Line."

This lesson was written using Git Bash 1.9.0 and the Windows 7 operating system. Equivalent file paths for OS X/Linux have been included where possible. Nonetheless, as commands and flags can change slightly between operating systems OS X/Linux users are referred to Deborah S. Ray and Eric J. Ray, "*Unix and Linux: Visual Quickstart Guide*", 4th edition (2009) which covers interoperability in greater detail.

The files used in this lesson are available on "Figshare". The data contains the metadata for journal articles categorised under 'History' in the British Library ESTAR database. The data is shared under a CC0 copyright waiver.

Download the required files, save them to your computer, and unzip them. If you do not have default software installed to interact with .zip files, we recommend 7-zip for this purpose. On Windows, we recommend unzipping the folder provided to your c: drive so the files are at `c:\proghist\` . However, any location will work fine, but you may have to adjust your commands as you are following along with this lesson if you use a different location. On OS X or Linux, we similarly recommend unzipping them to your user directory, so that they appear at `/user/USERNAME/proghist/` . In both cases, this means that when you open up a new terminal window, you can just type `cd proghist` to move to the correct directory.

Counting files

You will begin this lesson by counting the contents of files using the Unix shell. The Unix shell can be used to quickly generate counts from across files, something that is tricky to achieve using the graphical user interfaces (GUI) of standard office suites.

In Unix the `wc` command is used to count the contents of a file or of a series of files.

Open the Unix shell and navigate to the directory that contains our data, the `data` subdirectory of the `proghist` directory. Remember, if at any time you are not sure where you are in your directory structure, type `pwd` and use the `cd` command to move to where you need to be. The directory structure here is slightly different between OS X/Linux and Windows: on the former, the directory is in a format such as `~/users/USERNAME/proghist/data` and on Windows in a format such as `c:\proghist\data`.

Type `ls` and then hit enter. This prints, or displays, a list that includes two files and a subdirectory.

The files in this directory are the dataset `2014-01_JA.csv` that contains journal article metadata and a file containing documentation about `2014-01_JA.csv` called `2014-01_JA.txt`.

The subdirectory is named `derived_data`. It contains four .tsv files derived from `2014-01_JA.csv`. Each of these includes all data where a keyword such as `africa` or `america` appears in the 'Title' field of `2014-01_JA.csv`. The `derived_data` directory also includes a subdirectory called `results`.

Note: CSV files are those in which the units of data (or cells) are separated by commas (comma-separated-values) and TSV files are those in which they are separated by tabs. Both can be read in simple text editors or in spreadsheet programs such as Libre Office Calc or Microsoft Excel.

Before you begin working with these files, you should move into the directory in which they are stored. Navigate to `c:\proghist\data\derived_data` on Windows or `~/users/USERNAME/proghist/data/derived_data` on OS X.

Now that you are here you can count the contents of the files.

The Unix command for counting is `wc`. Type `wc -w 2014-01-31_JA_africa.tsv` and hit enter. The flag `-w` combined with `wc` instructs the computer to print a word count, and the name of the file that has been counted, into the shell.

As was seen in "Introduction to the Bash Command Line", flags such as `-w` are an essential part of getting the most out of the Unix shell as they give you better control over commands.

If your research is more concerned with the number of entries (or lines) than the number of words, you can use the line count flag. Type `wc -l 2014-01-31_JA_africa.tsv` and hit enter. Combined with `wc` the flag `-l` prints a line count and the name of the file that has been counted.

Finally, type `wc -c 2014-01-31_JA_africa.tsv` and hit enter. This uses the flag `-c` in combination with the command `wc` to print a character count for `2014-01-31_JA_africa.tsv` .

Note: OS X and Linux users should replace the `-c` flag with `-m` .

With these three flags, the most obvious thing historians can use `wc` for is to quickly compare the shape of sources in digital format - for example word counts per page of a book, the distribution of characters per page across a collection of newspapers, the average line lengths used by poets. You can also use `wc` with a combination of wildcards and flags to build more complex queries. Type `wc -l 2014-01-31_JA_a*.tsv` and hit enter. This prints the line counts for `2014-01-31_JA_africa.tsv` and `2014-01-31_JA_america.tsv` , offering a simple means of comparing these two sets of research data. Of course, it may be faster to compare the line count for the two documents in Libre Office Calc, Microsoft Excel, or a similar spreadsheet program. But when wishing to compare the line count for tens, hundreds, or thousands of documents, the Unix shell has a clear speed advantage.

Moreover, as our datasets increase in size you can use the Unix shell to do more than copy these line counts by hand, by the use of print screen, or by copy and paste methods. Using the `>` redirect operator you can export your query results to a new file. Type `wc -l 2014-01-31_JA_a*.tsv > results/2014-01-31_JA_a_wc.txt` and hit enter. This runs the same query as before, but rather than print the results within the Unix shell it saves the results as `2014-01-31_JA_a_wc.txt` . By prefacing this with `results/` it moves the .txt file to the `results` sub-directory. To check this, navigate to the `results` subdirectory, hit enter, type `ls` , and hit enter again to see this file listed within `c:\proghist\data\derived_data\results` on Windows or `/users/USERNAME/proghist/data/derived_data/results` on OS

X/Linux.

Mining files

The Unix shell can do much more than count the words, characters, and lines within a file. The `grep` command (meaning 'global regular expression print') is used to search across multiple files for specific strings of characters. It is able to do so much faster than the graphical search interface offered by most operating systems or office suites. And combined with the `>` operator, the `grep` command becomes a powerful research tool can be used to mine your data for characteristics or word clusters that appear across multiple files and then export that data to a new file. The only limitations here are your imagination, the shape of your data, and - when working with thousands or millions of files - the processing power at your disposal.

To begin using `grep`, first navigate to the `derived_data` directory (`cd ..`). Here type `grep 1999 *.tsv` and hit enter. This query looks across all files in the directory that fit the given criteria (the .tsv files) for instances of the string, or character cluster, '1999'. It then prints them within the shell.

Note: there is a large amount of data to print, so if you get bored hit `ctrl+c` to cancel the action. Ctrl+c is used to cancel any process in the Unix shell.

Press the up arrow once in order to cycle back to your most recent action. Amend `grep 1999 *.tsv` to `grep -c 1999 *.tsv` and hit enter. The shell now prints the number of times the string 1999 appeared in each .tsv file. Cycle to the previous line again and amend this to `grep -c 1999 2014-01-31_JA_*.tsv > results/2014-01-31_JA_1999.txt` and hit enter. This query looks for instances of the string '1999' across all documents that fit the criteria and saves them as `2014-01-31_JA_1999.txt` in the `results` subdirectory.

Strings need not be numbers. `grep -c revolution 2014-01-31_JA_america.tsv`

`2014-02-02_JA_britain.tsv` , for example, counts the instances of the string `revolution` within the defined files and prints those counts to the shell. Run this and then amend it to `grep -ci revolution 2014-01-31_JA_america.tsv 2014-02-02_JA_britain.tsv` . This repeats the query, but prints a case insensitive count (including instances of both `revolution` and `Revolution`). Note how the count has increased nearly 30 fold for those journal article titles that contain the keyword ‘revolution’. As before, cycling back and adding `> results/` , followed by a filename (ideally in .txt format), will save the results to a data file.

You can also use `grep` to create subsets of tabulated data. Type `grep -i revolution 2014-01-31_JA_america.tsv 2014-02-02_JA_britain.tsv > YEAR-MONTH-DAY_JA_america_britain_i_revolution.tsv` (where `YEAR-MONTH-DAY` is the date you are completing this lesson) and hit enter. This command looks in both of the defined files and exports any lines containing `revolution` (without regard to case) to the specified .tsv file.

The data has not been saved to to the `results` directory because it isn’t strictly a result; it is derived data. Depending on your research project it may be easier to save this to another subdirectory. For now have a look at this file to verify its contents and when you are happy, delete it using the `rm` command. *Note: the `rm` command is very powerful and should be used with caution. Please refer to “[Introduction to the Bash Command Line]” for instructions on how to use this command correctly.*

Finally, you can use another flag, `-v` , to exclude data elements when using the `grep` command. Type `grep -iv revolution 2014*_JA_a*.tsv > 2014_JA_iv_revolution.csv` and hit enter. This query looks in the defined files (three in total) and exports all lines that do not contain `revolution` or `Revolution` to `c:\proghist\data\derived_data\2014_JA_iv_revolution.csv` .

Note that you have transformed the data from one format to another - from .tsv to .csv. Often there is a loss of data structure when undertaking such transformations. To observe this for yourself, run `grep -iv revolution 2014*_JA_a*.tsv > 2014_JA_iv_revolution.tsv` and open both the .csv and .tsv files in Libre Office Calc, Microsoft Excel, or a similar spreadsheet program. Note the differences in column delineation between the two files.

Summary

Within the Unix shell you can now:

- use the `wc` command with the flags `-w` and `-l` to count the words and lines in a file or a series of files.
- use the redirector and structure `> subdirectory/filename` to save results into a subdirectory.
- use the `grep` command to search for instances of a string.
- use with `grep` the `-c` flag to count instances of a string, the `-i` flag to return a case insensitive search for a string, and the `-v` flag to exclude a string from the results.
- combine these commands and flags to build complex queries in a way that suggests the potential for using the Unix shell to count and mine your research data and research projects.

Conclusion

In this lesson you have learnt to undertake some basic file counting, to query across research data for common strings, and to save results and derived data. Though this lesson is restricted to using the Unix shell to count and mine tabulated data, the processes can be easily extended to free text. For this we recommend two guides written by William Turkel:

- William Turkel, 'Basic Text Analysis with Command Line Tools in Linux' (15 June 2013)
- William Turkel, 'Pattern Matching and Permuted Term Indexing with Command Line Tools in Linux' (20 June 2013)

As these recommendations suggest, the present lesson only scratches the surface of what the Unix shell environment is capable of. It is hoped, however, that this lesson has provided a taster sufficient to prompt further investigation and productive play.

For many historians, the full potential of these tools may only emerge upon embedding these skills into a real research project. Once your research grows, and, with it, your research data, being able to manipulate, count and mine thousands of files will be extremely useful. For if you choose to build on this lesson and investigate the Unix shell further you will find that even a large collection of files which do not contain any alpha-numeric data elements, such as image files, can be easily sorted, selected and queried in the Unix shell.

About the authors

James Baker is a Curator in the Digital Research team at the British Library and an historian of eighteenth century Britain.   Ian Milligan is an assistant professor of history at the University of Waterloo.

Suggested Citation

James Baker and Ian Milligan , "Counting and mining research data with Unix," *Programming Historian*, (2014-09-20), http://programminghistorian.org/lessons/research-data-with-unix