HW04 - Bash commands

Stat 133, Fall 2016, Prof. Sanchez

This assignment has 4 main purposes:

- practicing with the command line
- navigating the filesystem and managing files
- using redirections and pipes
- practice basic manipulation of data files

Basic Bash shell commands

The first part of the lab involves navigating the file system and manipulating files (and directories) with the following basic bash commands:

- `pwd`: print working directory
- `ls`: list files and directories
- `cd`: change directory (move to another directory)
- `mkdir`: create a new directory
- `touch`: create a new (empty) file
- `cp`: copy file(s)
- `mv`: rename file(s)
- `rm`: delete file(s)

If you are using git-bash you don't have the `man` command to see the manual documentation of other commands. In this case you can check the *man* pages online:

http://man7.org/linux/man-pages/index.html

Write your commands in a text editor (NOT a word processor) and save them in a text file called `stat133-hw4-first-last.txt` where `first` and `last` are your first and last names (e.g. `stat133-hw4-gaston-sanchez.txt`):

Part 1

- Create a new directory `stat133-hw4`
- Change to the directory `stat133-hw4`
- Use the command `curl` to download the following text file:

```
# the option is the letter O (Not the number 0)
curl -O http://textfiles.com/food/bread.txt
```

- Use the command `ls` to list the contents in your current directory
- Use the command `curl` to download these other text files:
  - http://textfiles.com/food/btaco.txt
  - http://textfiles.com/food/1st_aid.txt
  - http://textfiles.com/food/beesherb.txt

- Use the command `curl` to download the following csv files:
  - http://archive.ics.uci.edu/ml/machine-learning-databases/forest-fires/forestfires.csv
  - http://www.math.uah.edu/stat/data/Fisher.csv
  - http://web.pdx.edu/~gerbing/data/cars.csv
- Now try `ls -l` to list the contents in your current directory in long format
- Inside `stat133-hw4` create a directory `data`
- Change to the directory `data`
- Create a directory `txt-files`
- Create a directory `csv-files`
- Use the command `mv` to move the `bread.txt` file to the folder `txt-files`:
- Use the wildcard `*` to move all the text files to the directory `txt-files`
- Use the wildcard `*` to move all the `.csv` files to the directory `csv-files`
- Go back to the parent directory `stat133-hw4`
- Create a directory `copies`
- Use the command `cp` to copy the `bread.txt` file, that is in the folder `txt-files`, to the `copies` directory
- Copy all the `.txt` files in the directory `copies`
- Copy all the `.csv` files in the directory `copies`
- Change to the directory `copies`
- Use the command `mv` to rename the file `bread.txt` as `bread-recipe.txt`
- Rename the file `Fisher.csv` as `iris.csv`
- Rename the file `btaco.txt` as `breakfast-taco.txt`
- Change to the parent directory (i.e. `stat133-hw4`)
- Rename the directory `copies` as `copy-files`
- Find out how to use the `rm` command to delete the directory `copy-files`
- List the contents of the directory `txt-files` displaying the results in reverse (alphabetical) order

Optional challenge: If you are already familiar with the basic commands to navigate the filesystem (or if you want to expand your R skills), use the R commands to manipulate files and directories to perform the exact same tasks from within R. See `?files` for more information.

- `getwd()`
- `setwd()`
- `download.file()`
- `dir.create()`
- `list.files()`
- `list.dirs()`
- `file.create()`
- `file.copy()`
- `file.rename()`

- `file.remove()`

Redirection, Pipes, and other commands

In addition to learning the commands for navigating your filesystem, you should also learn about other basic unix utilities to do some data-file manipulation. In this section you will be working the data set `cpds.csv` that is available in the github repository of the course:

https://raw.githubusercontent.com/ucb-stat133/stat133-fall-2016/master/data/cpds.csv

Open a terminal emulator (command line), and type the following commands:

```
# create a new directory and 'cd' into it
mkdir pipelines
cd pipelines

# download data file
curl -O https://raw.githubusercontent.com/ucb-stat133/stat133-fall-2016/master/data/cpds.csv

# list available files
ls
```

Use `head` and `tail` to take a look at the first and last lines of the CSV file

```
head cpds.csv
tail cpds.csv
```

Extracting Columns with `cut`

To pull out vertical columns from a file you can use the `cut` command. This Unix utility operates based either on character position within the column when using the `-c` option, or on delimited fields when using the `-f` option.

Options for the `cut` command:

- `-f 1,3` Returns columns 1 and 3, delimited by tabs
- `-d ","` Use commas as the delimiters, instead of tabs; this option is used in conjuction with the `-f` option
- `-c 3-8` Returns characters 3 through 8 from the file or stream of data

With the `-c` option, numbers are given to indicate which characters to extract; with the `-f` option, the numbers indicate which columns to extract; the `-d` option indicates the type of field delimiter. By default `cut` expects tabs as the delimiter. So, for instance, to indicate a comma as a delimiter, use `-d ","`

To pull out the first column `year` from the CSV file, you need to specify `-f 1` followed by character-delimiter flag `-c ";"`

```
# first column
cut -f 1 -d "," cpds.csv
```

You can just simply look at the first five lines piping the previous command with the `head` command:

```
# first lines of first column
cut -f 1 -d "," cpds.csv | head -n 5
```

To subset the second to fourth columns, and "save" them in a new file, you use the redirection output operator `>`:

```
cut -f 2-4 -d "," cpds.csv > columns-2-4.csv
```

Sorting lines with `sort`

You can use the `sort` command to sort the lines of a file, or the input passed to `sort`. By default, `sort` starts with the first character of the line and the first column of data:

```
ls | sort

cut -f 1 -d "," cpds.csv | sort | tail -n 5
```

Options for the `sort` command

- `-n` Sort by numeric value rather than alphabetically
- `-r` Sort in reverse order, z to a or high numbers to low numbers
- `-k 3` Sort lines based on column 3, with columns delimited by spaces or tabs
- `-t ","` Use commas for delimiters, instead of the default or tabs or white spaces
- `-u` Return only a single unique representative of repeated items

To sort based on other columns (whether separated by tabs or spaces), use the `-k` option followed by the number of the column you wish to use for sorting. Note that because values are sorted in ASCII order, blanks come alphabetically before the letter A. Another behavior is that all capital letters come before lowercase letters, so capital `Z` is alphabetically before lowercase `a`.

Sorting can proceed numerically instead of alphabetically when the `-n` option is used.

Isolating unique lines with `uniq`

Another powerful and frequently used command for extracting a subset of values from a file is `uniq`. This command removes consecutive identical lines from a file, leaving one unique representative. I order to be removes, the marching lines have to occur in immediate succession, without any intervening different lines. To get a single representative of each unique line from the entire file, in most cases you would need to first sort the lines with the `sort` command to group matching lines together.

The `uniq` command can be used with the `-c` option to count the number of occurrences of a line or value.

Options for the `uniq` command:

- `-c` Counts the number of occurrences of each unique line
- `-f 4` Ignore the first 4 fields (columns delimited by any number of spaces) in determining uniqueness
- `-i` Ignores cases when determining uniqueness

For example, you can use `uniq` to extract the unique values of years:

```
cut -f 1 -d "," cpds.csv | sort | uniq
```

If you want to redirect the output to a file `years.csv`, then use the redirection operator `>`:

```
cut -f 1 -d "," cpds.csv | sort | uniq > years.csv
```

Extracting particular rows from a file

What if you want to extract those line in `cpds.csv` starting with the value `2010`? The command `grep` is a tool that quickly extracts only those lines of a file that match a particular regular expression.

```
grep "2010" cpds.csv
```

The first argument, `"2010"`, is the regular expression, and the second argument specifies the source file you want it to examine. The `grep` program scans the file and displays only those lines that contain the search phrase. You need to use quotes around the regular expression as a good practice.

In the previous example, the results were simply sent to the screen. But you can redirect them to a file `cpds-2000.csv`

```
grep "2000" cpds.csv > cpds-2000.csv
```

Now you have a file that is a subset of the original, containing only those lines with year `"2000"`. The only issue now is that the new file doesn't have a header. To solve this, you can do something like this:

```
# redirect the header
head -n 1 cpds.csv > cpds-2000.csv

# append the lines with year "2000"
grep "2000" cpds.csv >> cpds-2004.csv
```

Options that modify the behavior of `grep`

- `-c` Show only a count of the results in the file
- `-v` Invert the search and show only lines that do NOT match
- `-i` Match without regard to case
- `-E` Use regular expression syntax "Extended" regex
- `-l` List only the file names containing matches
- `-n` Show the line numbers of the match
- `-h` Hide the filenames in the output

Your Turn

Here you will work through a few examples to start giving you a feel for using the bash shell to manage your workflows and process data.

Your first mission: Extracting columns

- use `cut` to extract the second column of `cpds.csv`
- use `cut` to extract the second column of `cpds.csv` and pipe it with `less` to see the outputs with the *paginator*
- use `cut` to extract the second column of `cpds.csv` and pipe it with `head` to look at the first 5 lines
- use `cut` to extract the second column of `cpds.csv` and pipe it with `tail` to look at the last 3 lines
- use `cut` and `uniq` to display the names of the countries
- use `cut` and `uniq -c` to display the counts of the countries

Your second mission: Identifying patterns and subsetting lines

- Use `grep` to display those lines of `cpds.csv` for `year` 1960
- Redirect the previous command to a csv file `cpds-1960.csv`
- Use `grep` to display those lines of `cpds.csv` for `country` USA
- Redirect the previous command to a csv file `cpds-usa.csv`
- Use `grep` to display those lines of `cpds.csv` for years 1960 and 1970
- Use `grep` to display those lines of `cpds.csv` in which the country name begins with the letter `"S"` (e.g. Spain, Sweden, Switzerland)
- Write a command that displays the values for columns `year`, `country` and `unemp`, for year 1960, and sorted by `unemp` in ascending order. Here are the first 6 lines for the required output:

  ```
  1960,"Switzerland",0.05
  1960,"Luxembourg",0.09
  1960,"New Zealand",0.12
  1960,"Netherlands",0.72
  1960,"Germany",1.03
  1960,"Norway",1.2
  ```

- Write a command that displays the number of lines with missing values `NA` in column `"outlays"`, for records in `Iceland`. Your answer should be 10.
- Write a command that displays the number of lines with negative values in column `"realgdpgr"`, for records in `Iceland`. Your answer should be 9.