Useful Linux Commands for Large Data Sets

If you have installed CYGWIN correctly with the "chere" option as instructed, when you right click in a windows file explorer window (showing the contents of a folder) you will see a menu entry "Bash prompt here." This will open a bash shell (command interpreter) in a new window whose current working directory is the same as the folder that is open in the file explorer window.

The following Linux commands are very useful in the context of dealing with large files on windows:

`pwd`

Print working directory: prints the current working directory. Useful for making sure that you are in the folder you intend to be.

`ls`

List: lists the files in the current folder.

`ls –a`

Lists out all of the files in the current folder including "hidden" files that begin with a period (".").

`wc –l ProjectTrainingData.csv`

Word count: With the –l option, wc counts the number of lines in a file (in this case the file ProjectTrainingData.csv). Useful for figuring out how many records are in a file.

`less ProjectTrainingData.csv`

A text-based pager for reading through a file. This works with files of any size. Hitting the space bar gives you the next page. Typing "q" quits and exits less. Typing "b" takes you back a page. See the manual page for less for more possibilities (i.e., man less). Note that man pages its output with less. (Another similar pager was called "more" so that explains the name "less".)

`head ProjectTrainingData.csv`

Print out the first 10 lines of the file ProjectTrainingData.csv

`head –n 100 ProjectTrainingData.csv`

Print out the first 100 lines of the file ProjectTrainingData.csv.

```
head –n 1000000 ProjectTrainingData.csv > ProjectTrainingData1-1MM.csv
```

Save the first 1 million lines of ProjectTrainingData.csv into the file ProjectTrainingData1-1MM.csv. Very useful for creating smaller data sets for code development purposes.

```
tail ProjectTrainingData.csv
```

Print out the last 10 lines of the file ProjectTrainingData.csv.

```
tail –n 100 ProjectTrainingData.csv
```

Print out the last 100 lines of the file ProjectTrainingData.csv.

```
tail –n +2 ProjectTrainingData.csv
```

Print out the file starting at line 2. This can be used to skip things like a line of column labels. Compare the difference between

```
tail –n +1 ProjectTrainingData.csv | less
tail –n +1 ProjectTrainingData.csv | less
```

```
|
```

This is the vertical bar (generally on the key above the Enter key). It allows you to route the output of one command as the input of another command. This is called piping and | is referred to as a pipe. For example

```
head –n 1000000 ProjectTrainingData.csv | tail –n 1000 | less
```

allows you to page through the last 1,000 lines of the first 1,000,000 lines; that is, page through lines 999,001 to 1,000,000

```
man head
```

Manual: Show the manual page for the command head. Of course man works for other commands as well and is how you get the “official” documentation for Linux commands.

```
split –l 1000000 –d –a 3 --additional-suffix=.csv ProjectTrainingData.csv Train-
```

Split splits up files. This command splits the file ProjectTrainingData.csv into 1000000 line chunks that have files names starting with Train-000.csv, Train-001.csv, etc. The option –l specified the number of lines for each file, -d says number then with integers, -a 3 indicates three places (i.e., 000, 001, etc.), --additional-suffix=.csv adds the .csv file extension to the output files, ProjectTrainingData.csv is the source file used, and Train- is the main body of the output file name. Note that the column labels will be in the first file. The file name ProjectTrainingData.csv can be substituted by a hyphen (“-“) to cause the

command to read from the standard input. Thus, to skip the column labels in the first file, you can use the command

```
tail –n +2 ProjectTrainingData.csv | split –l 1000000 –d –a 3 --
additional-suffix=.csv - Train-
```

Note that the command above should be all on one line, but it is wrapped by word. For more details about split, see the manual page or do an internet search.

```
shuf ProjectTrainingData.csv >  TrainPerm.csv
```

shuf creates a random permutation of the lines of the file ProjectTrainingData.csv and writes it out to the file TrainPerm.csv. The commands shuf and split can be used to create random subsamples of large files.

```
cut –d , -f 3 ProjectTrainingData.csv
```

Cut cuts out parts of a file. It can be used to pull out specific fields (i.e., variables or features). The –d , option sets the field delimiter to ",". The –f 3 option pulls out the third field. The option –f 2-4 pulls out fields 2, 3, and 4. The option –f 2- pulls out fields 2 to the end of the line and –f -4 pulls out fields 1 to 4.

```
sort –t , -k 6 ProjectTrainingData.csv > TrainSortSiteID.csv
```

Sort the input file. The option –t , sets the field delimiter to "," and the option –k 6 means sort by the 6th field (the site ID). This 6th field is treated as a character string. The command

```
sort –n  –t , -k 3 ProjectTrainingData.csv > TrainSortSiteID.csv
```

does a numeric sort (i.e., treats the field as a number) on the time field. Note that I have not excluded the column labels from these sorts. So

```
tail –n +2 ProjectTrainingData.csv | sort –n  –t , -k 3 - > TrainSortSiteID.csv
```

would probably make more sense. Here "-" is substituted for the file name in the sort command to make it read from the standard input.

The sort command is very powerful, but somewhat complex. So take a look at the manual page or search for more information on the internet. The Linus sort is an on disk sort, so it never reads the whole file into memory. As a result, it can sort very large files.