Command Lines for Basic Data Management and Text Processing

Weimao Ke w/ ChatGPT Assistance

Command Lines for Text and Statistics

Introduction Shell and Command Line

- What is a Shell?
- What is a Command Line?
- Importance and applications of command line
  A shell is a computer program with a command line interface (no graphics) which allows you to control your computer using commands entered with a keyboard.
- Compare to GUI with graphics and mouse clicks
- Can be scripted and replicated.

Brief Background of Command Lines

- Evolution of command line interfaces
  - MS DOS, Windows CMD, PowerShell, WSL
  - Unix, Linux, and MacOS (BSD) terminals
- Different types of shells: sh, bash, zsh, etc.
- Use of command lines on different OS:
  - On MacOS or Linux, use the Terminal app.
  - On Windows 11, the best is to use Windows Terminal:
  - https://apps.microsoft.com/store/detail/windows-terminal/9N0DX20HK701?hl=en-us&gl=us&rtc=1

Today's Focus: Preprocessing Without Programming

- Popular commands for text processing and basic statistics
- How to combine the power of multiple tools
- Complex operations without complex program structure, e.g. no loops
- Examples based on commands the Mac/Linux (`bash`)

3. Command Line Environment

- Command line prompt, e.g. a leading `$` sign.
- Basic syntax of a command: `command_name [arguments] [files]`

Example:

```
$ head -1000 big_data.txt
```

Navigating Files and Directories

- `pwd`: print working directory
- `cd`: change directory
- `ls`: list contents of directory

Examples

```
$ pwd
/Users/username/Desktop
$ ls
file1.txt file2.txt authors/
$ cd authors
$ pwd
/Users/username/Desktop/authors
$ ls -l
[verbose list of files]
```

5. Creating and Removing Files/Directories

- `touch`: create a file
- `mkdir`: make directory
- `rm`: remove file/directory

Examples

```
$ touch new.txt
$ rm old.txt
$ mkdir appendix
```

Cat, Head, Tail

- `cat`: display entire file
- `head`: display top part of file
- `tail`: display bottom part of file

Examples:

```
$ cat article.txt
$ tail -50 article.txt
```

String Translation

`tr` – translate characters

```
tr '[:upper:]' '[:lower:]'
```

Count

- `wc` to count the number of words, lines, characters, and bytes

Example:

```
$ wc -l records.csv         # count the number of lines/rec
$ wc -m shakespeare.txt     # count the number of words
```

Aggregation and Duplicate Removal

- `uniq` to merge duplicates and count them
- Data example of `votes.txt`:

```
Messi
Messi
Messi
Messi
CR7
CR7
```

Example to count votes

```
$ uniq -c votes.txt
```

Assume votes are already sorted.

Sort: Ordering Text or Numbers

- Basics of `sort`
- Practical examples of sort usage

```
$ sort names.txt        # sort a list of names in alphabeti
$ sort -nr scores.txt   # sort a list of numeric scores in
```

Grep: Searching and Filtering

- Basics of `grep`: to search and filter text lines
- Practical example of grep usage

```
$ grep 'Calpurnia' shakespeare.txt
```

Input, Output, Redirection, and Pipes

- Understanding standard input, output, and error
- Redirecting output (with > and >>)

```
$ cat in.txt              # read and print data to screen (
$ cat in.txt > out.txt    # redirect the output to a file
$ grep 'fellows' more.txt >> fellowship.txt  # redirect out
```

Pipes and Output-Input Pipeline

Piping (with |) is a powerful mechanism that:

- allows the output of one command to be input for another;
- enables useful combination of related tools

```
$ cat shakespeare.txt | grep 'Caesar' | wc -l   # count the
$
```

Advanced Text / Data Processing

- Importance of text processing in data analysis
- Command line tools for data gathering and text processing

Data Gathering & Communication Commands

- `wget`: downloading files
- `curl`: getting and sending data from/to servers

```
$ wget "https://raw.githubusercontent.com/karpathy/char-rnn
```

Awk: Text Processing Power Tool

- Basics of `awk`: pattern-directed scanning and processing
- Practical examples of awk usage

```
cat records.csv | awk -F, '{print $1,$3,$5}'    # select/ou
```

Perl with Inline Search and Text Processing

- Perl: a programming language for text processing
- Perl with inline regex `perl -pe ...`
- Practical examples of perl usage

```
cat article.txt | perl -pe 's/iSchool/CCI/g'
```

Regular Expression

A regular expression is

a sequence of symbols that forms a search pattern.

- Symbols include alphabets, numbers, and special characters.
- They can be used for text search/matching, or search & replace.

Basic Regex Syntax

Search pattern:

`/pattern/modifiers`

Modifier: + `i` for case-insensitive matching

+ `g` for a global match (find all matches instead of first match)

+ `m` for multiline matching

Example, search for `LEADING` case-insensitive:

`/LEADING/i`

Regex Patterns

Meta Characters:

- `.` for any single character
  - `\.` for a dot (period) symbol
- `\w` for a word character or alphabet, `a`, `b`, . . .
- `\d` for a digit, e.g. `0`, `1`, . . .
- `\s` for a whitespace character
- `\b` for word boundary, e.g. space, punctuation
- `uxxxx` for a unicode character
- `[:punct:]` for punctuation

Regex Patterns

Number of occurrences:

- `?` for 0 or 1 occurrence
- `*` for 0 or more occurrences
- + for at least 1 occurrences

Reference to matched patterns:

- (pattern1) (pattern2) (pattern3) ...

can be referred to as $1, $2, $3, ...

Regex Pattern Example

Search and replace:

```
s/(\w+)@(\w+).edu/$1 from $2/g
```

This changes an email address like `john@drexel.edu` to `john from Drexel`.

Case Study: Exploratory Analysis of Shakespeare's Plays

Shakespeare's Plays, to be downloaded from:
https://www.kaggle.com/datasets/kingburrito666/shakespeare-plays
which includes:

1. `Shakespeare_data.csv`
2. `alllines.txt`

Case Study: Exploratory Analysis of Shakespeare's Plays

Tasks:

1. Count the # lines, # plays, and # lines per play
2. Identify unique players and count them in each play
3. Search for play containing certain keywords, e.g. `Brutus`
4. Text processing and term statistics

Essential Command Line Shortcuts

- Command history
  - `history` to show the list of used commands
  - Up (back) and Down (forward) to go back to a previous command
- Tab completion:
  - type partial command name or file name
  - press `Tab` to complete
- Command editing shortcuts:
  - `Ctrl+A` to the beginning
  - `Ctrl+E` to the end

Command Line Best Practices

- Command organization and clarity
- Utilizing man pages effectively
  - i.e. enter `man name_of_command`
- Keeping the system safe while using command line
  - Don't use `sudo` (super user privileges) unless you know what it entails!
- In the age of LLMs, use tools like ChatGPT to help you with commands/code.

The Power of Command Line

- Know the commands and what they are good at
- Find a way to combine them with | (piping)
- Always sample and test first, before a run with the full dataset