Data Engineering - Lecture 5

UNIX recap

Shamindra Shrotriya (CMU)

Some success stories of data engineering

Apache Hadoop

Distributed large scale processing

Inspired by the map-reduce framework (Google)

Apache Kafka

Large scale **streaming** data

Developed at Linkedin (handle newsfeed analytics)

Adopted by Twitter

Apache Airflow

Large scale machine learning **pipelining**

Developed by **Airbnb**

Do we need to learn all these tools to be a data-engineer?

*Is there an **alternative** structured way to approach learning these these data-eng principles, and deeply imbibe them in our daily workflow?*

Definitely - we just need to **travel back in time** to the **present**!

We should go back and learn **`UNIX`**, **`SQL`**, **`tmux`**, **`Make`**, etc

Takeaway: Developed **over past six decades**, and **still going strong today**!

What common principles do these tools share though?

Highly **extensible** (programmable) systems

Easily **configurable** - just send me the **config** file!

Structured approach to **pipelining systems**

Consistent **grammar** (“self-documenting”)

Nope! Command line + GUIs = 💙

Our **primary goal** is to become a **productive** and **happy** data engineer/scientist

Use the best tool for the given task!

Does your task involve a lot of animation, **graphic** previews, visual demos? **GUI!**

Does your task involve a lot of **text** driven processing

> file navigation, manipulation, previews, searching, replacing? **Command line**

Takeaway: using both GUI/UNIX appropriately will improve your work productivity!

command **prompt is hard to navigate**, any easier way?

Sure - keyboard shortcuts can simplify prompt navigation

`Ctrl + a`	go to the start of the prompt
`Ctrl + k`	clear typed contents from cursor till end of line
`Ctrl + l`	clear screen
`Ctrl + u`	clear typed contents
`Ctrl + w`	clear previous word

Can we quickly ***retrieve*** a command from our `history`?

Indeed - `Ctrl + r` to for `r`everse history search

`Ctrl + r`

New prompt appears, waiting for you to start reverse searching

This gets even cooler with fuzzy finding (`fzf`), where search typos are forgiven

We’ll learn more about this next week

Key idea `command`: *`text`* → *`text`*

The command line can be thought of as an

advanced text processing language

Takeaway: text is ***the*** **universal interface** for both input/output in the command line

Can we ***combine*** commands together nicely?

Yep - we can chain command output input using `|` operator

Syntax ***`command1`*** `|` ***`command2`***

The `|` takes the **output** of ***`command1`*** and **sends it as input** to ***`command2`***

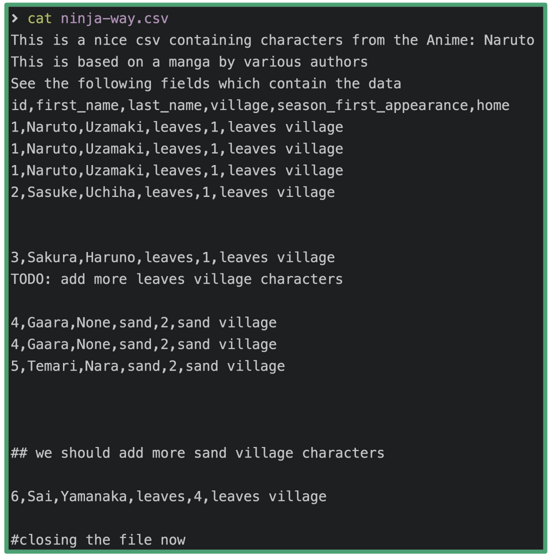
Called the **pipe operator**, remind you of something? Yep `%>%` in R!

Can read the pipe (`|`) as the words “and then”, just like we did in R

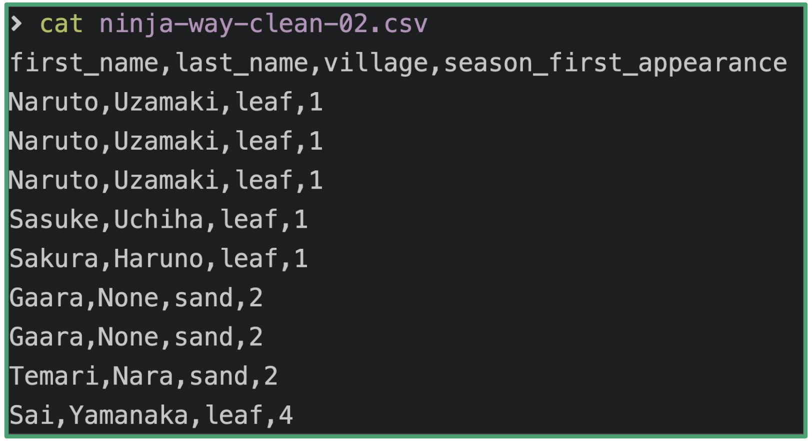
Takeaway: The pipe provides a grammar for function composition in UNIX

So what did all our text processing work achieve?

We started with **`ninja-way.csv`** and ended with **`ninja-way-clean-02.csv`**



```
> cat ninja-way.csv
This is a nice csv containing characters from the Anime: Naruto
This is based on a manga by various authors
See the following fields which contain the data
id,first_name,last_name,village,season_first_appearance,home
1,Naruto,Uzamaki,leaves,1,leaves village
1,Naruto,Uzamaki,leaves,1,leaves village
1,Naruto,Uzamaki,leaves,1,leaves village
2,Sasuke,Uchiha,leaves,1,leaves village


3,Sakura,Haruno,leaves,1,leaves village
TODO: add more leaves village characters

4,Gaara,None,sand,2,sand village
4,Gaara,None,sand,2,sand village
5,Temari,Nara,sand,2,sand village



## we should add more sand village characters

6,Sai,Yamanaka,leaves,4,leaves village

#closing the file now
```



```
> cat ninja-way-clean-02.csv
first_name,last_name,village,season_first_appearance
Naruto,Uzamaki,leaf,1
Naruto,Uzamaki,leaf,1
Naruto,Uzamaki,leaf,1
Sasuke,Uchiha,leaf,1
Sakura,Haruno,leaf,1
Gaara,None,sand,2
Gaara,None,sand,2
Temari,Nara,sand,2
Sai,Yamanaka,leaf,4
```

Takeaway: All of this pre-processing was done without leaving the command line!

`sed + awk` give clean reproducible pipelines

We used **`sed`** to create **`ninja-way-clean-01.csv`**

We can just now run this through our **`awk`** pipeline

```
awk -F',' -v OFS="," '{ $1=$NF=""; print }' ninja-way-clean-01.csv | \

awk '!visited[$0]++' | \

sed 's/^,//g' | \

sed 's/,$//g' > \

Ninja-way-clean-02.csv
```

You can use this nice **`awk`** example guide and incorporate it into your workflow

Some more **fun use cases** of pipes

Modern: building mini apps using `fuzzy finder`

`fzf` is a remarkable utility to fuzzy find files by name.

```
> find . -type d | \
fzf --multi --height=80% --border=sharp --preview='tree -C
{}'
```

We just created a directory tree browsing **app** in **one line of code** (see: source)

Takeaway: `fzf` is an **indispensable** tool for **interactive search**

A reminder as to why I use the command line

I like using the command line because it’s ***fun***

Specifically it allows me to directly have a ***conversation*** with my **operating system**

References

Shotts, William (2019). *The Linux command line: a complete introduction.* No Starch Press [Link]

Evans, Julia *Bite Size Bash!* [Link]

Hogan, Brian (2021). *Small, Sharp Software Tools: Harness the Combinatoric Power of Command-Line Tools and Utilities.* Pragmatic Bookshelf [Link]

Janssens, Jeroen (2021). *Data Science at the Command Line.* O'Reilly Media, Inc. [Link]