Data Engineering - Lecture 2

Getting **comfortable** with the UNIX philosophy

Shamindra Shrotriya (CMU)

So ***where*** were we?

What are the driving principles of data engineering tools?

Highly **extensible** (programmable) systems

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

Structured approach to **pipelining systems**

Systematic **specification** of **dependencies**

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

Parallel + **distributed** processing

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

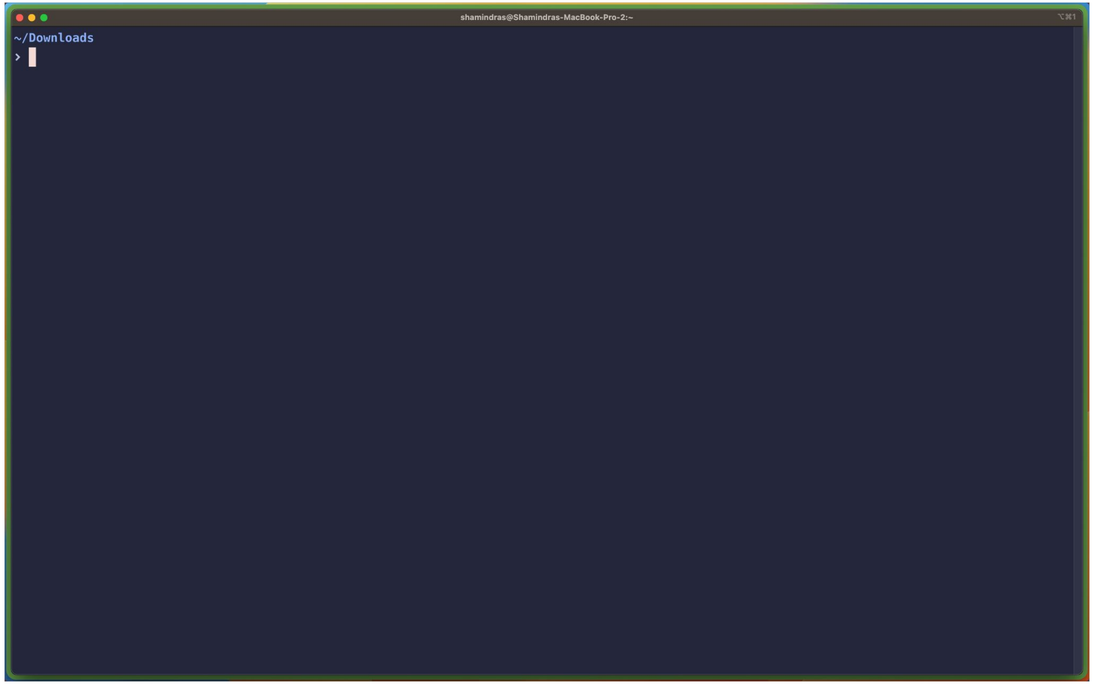
Is there an ***alternative*** *structured way to approach learning these these data engineering principles, and* ***deeply internalize*** *them in our* ***daily workflow****?*

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!

Starting UNIX: The terminal and the Shell



```
~/Downloads
>
```

Takeaway: We use a language called **bash** to enter our commands at the prompt

Let’s emulate basic operations we typically do via a GUI

Navigation

Manipulating files/directories

Inspecting **contents**

…

Searching through files/directories and their contents

Recap: viewing files and directories in UNIX

How do we list all files in the active directory?	`> ls`
Include hidden (“.”, dot) files?	`> ls -a`
Include metadata, e.g., date mod?	`> ls -l`
Make file sizes human readable?	`> ls -h`
All of the above?	`> ls -ahl`

Recap: fast directory navigation in UNIX

Change to HOME directory?	`> cd ~`
Change back to previous directory?	`> cd -`
Change to parent directory	`> cd ..`
Documentation on cd?	`> man cd`
Show directory tree with 2 levels of nesting	`> tree -L 2`
Print the working directory	`> pwd`

Recap: manipulating files/directories in UNIX

Copy `.bashrc` to `~/.bashrc`	`> cp .bashrc ~/.bashrc`
Move `.bashrc` to `~/.bashrc`	`> mv .bashrc ~/.bashrc`
Move and rename `.bashrc` to `~/.bashrc2`	`> mv .bashrc ~/.bashrc2`
Rename `.bashrc` to `.bashrc2`	`> mv .bashrc .bashrc2`
Make a nested subdir `./data/raw`	`> mkdir -p data/raw`

Recap: file/content viewing in UNIX

Get the line count for `~/.bashrc`	`> wc -l .bashrc`
Get the word count for `~/.bashrc`	`> wc -w .bashrc`
Interactively inspect `~/.bashrc` in pager	`> less ~/.bashrc`
Output contents of `schedule.csv`	`> cat schedule.csv`
View top 5 rows of `schedule.csv`	`> head schedule.csv`
View bottom 5 rows of `schedule.csv`	`> tail schedule.csv`

Is it the command line ***vs.*** GUIs?

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!

Some additional ***useful*** bash commands

`history`: storing our command history for easy review

Answer to: can we see all* the commands we’ve previously typed in bash?

```
> history
```

Note: It typically ignores the calls to the `history` command itself :)

Key: let bash keep track, and treat `history` like an on-demand file for your review

`less`: interactively inspect a file

Answer to: can we pull up file contents and interact with them (searching etc)?

```
> less file1.Rmd
```

“ephemeral” paginated print out contents of `file1.Rmd`

Once you press “q”, the print out is closed screen space is freed up again

Key: `less` **discourages context-switching** away from the terminal!

`sort`: `sort` contents of a (text) file

Answer to: can alphanumerically sort the contents of a text file?

```
> sort ~/temp.txt
```

Sorts a file in ascending (alphabetical) order

```
> sort -r ~/temp.txt
```

Sorts a file in **r**everse (alphabetical) order

```
> sort -n ~/temp.txt
```

Sorts a file in ascending (numeric) order

`find`: `find` files or directories

Answer to: can we quickly filter files of a given type?

```
> find . -type f -name '*.R'
```

Finds all `R` files in the current directory

```
> find ~ -type d -iname '*lib*'
```

Find directories matching a given name, in case-insensitive mode

```
> find root_path -maxdepth 2 -size +500k -size -10M
```

Find files matching a given size range, limiting the recursive depth to "2"

Natural concerns you may have

Too much typing can’t we minimize this?

The command **prompt is hard to navigate** with L/R arrows, any easier way?

I forgot that cool command from last week, can I **quickly retrieve** it?

Can we easily run all of these commands on **multiple files** instead of one?

I can see some of these commands being useful, but can we **combine** them?

This is **too much typing**, is there a way to minimize this?

Yes - aliases to the rescue!

```
> alias ll='ls -l'
```

Save in `~/.bashrc` and reload your terminal, and **then** type `ll`

```
> alias l='ls'

> alias lh='ls -h'

> alias lah='ls -ah'

> alias lla='ls -ahl'
```

Keep going - use **pneumonics**, and keep them 3 characters or less

Some more fun aliases to save those precious keystrokes

```
> alias ..='cd ..'; alias ...='cd ../..';

> alias md='mkdir -p'

> alias c='clear'

> alias t1='tree --level=1'; alias t2='tree --level=2';
```

Takeaway: for persistent aliases, store them in `~/.bashrc` and reload terminal

brace expansion - giving existing commands new powers

Answer to: can we use **sequences** to generate new text/files/directories?

```
> echo {01..11}
```

```
01 02 03 04 05 06 07 08 09 10 11
```

This is looping in a **succinct** format, i.e., ‘syntactic sugar’

```
> echo {a..f}
```

```
a b c d e f
```

Works with lower(upper) case letters too

`brace expansion` - existing commands get new powers

```
> touch slides-{01..04}.Rmd
```

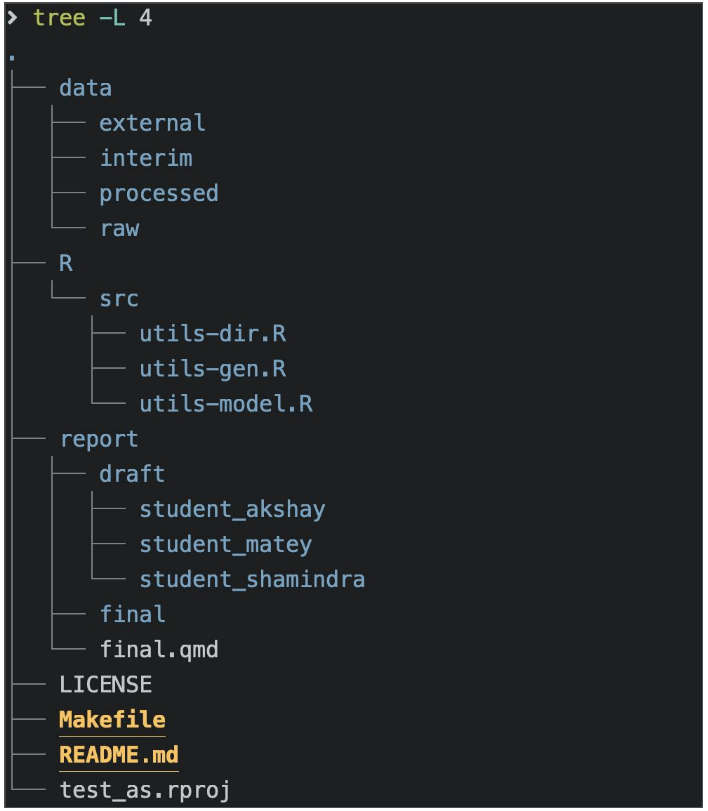
creates files! 01-slides.Rmd 02-slides.Rmd 03-slides.Rmd 04-slides.Rmd

```
> mkdir -p analysis_{ahmed,pratik,natalia,yue}
```

creates subdirs! analysis_ahmed/, ... , analysis_yue/

```
> mkdir -p data/{external,interim,processed,raw}
R/src/{utils-gen.R,utils-dir.R,utils-model.R}
report/{final,draft/student_{akshay,shamindra,matey}}; touch
README.md LICENSE Makefile report/final.qmd test_as.rproj;
```

`brace expansion` - existing commands get new powers



```
> tree -L 4
.
├── data
│   ├── external
│   ├── interim
│   ├── processed
│   └── raw
├── R
│   └── src
│       ├── utils-dir.R
│       ├── utils-gen.R
│       └── utils-model.R
├── report
│   ├── draft
│   │   ├── student_akshay
│   │   ├── student_matey
│   │   └── student_shamindra
│   ├── final
│   └── final.qmd
├── LICENSE
├── Makefile
├── README.md
└── test_as.rproj
```

```
> mkdir -p data/{external,interim,processed,raw}
R/src/{utils-gen.R,utils-dir.R,utils-model.R}
report/{final,draft/student_{akshay,shamindra,ma
tey}}; touch README.md LICENSE Makefile
report/final.qmd test_as.rproj;
```

< Produces this entire directory structure

Brace expansions are amazing - use em'!

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

Can we run a command on ***multiple files*** of the **same** type?

Globs to the rescue!

```
> ls *.Rmd
```

Wildcard **`list`** out all Rmd files

```
> wc -l *.(Rmd|html)
```

Line count all out all Rmd and html files

```
> cat *.Rmd
```

Concatenate all Rmd files and output to screen

Can we ***combine*** commands together nicely? Next week :)