AI VIETNAM
@aivietnam.edu.vn

Unix/Linux for Data Science

Lecturer: Nguyen Thai Ha (Ph.D)

Supporter: Nguyễn Thọ Anh Khoa (Ph.D Candidate)

Date: 21/06/2025 (Sat)

AIO2025 | **Link Source Code**

@AI VietNam

Learning Objectives

1 Nắm vững nền tảng Unix/Linux

Hiểu triết lý Unix, cấu trúc thư mục, và shell - công cụ thiết yếu cho phân tích dữ liệu

2 Thành thạo lệnh terminal cần thiết

Từ thao tác file cơ bản đến xử lý text, grep, awk, sed và các công cụ xử lý dữ liệu theo đường ống (pipe)

3 Khai thác và xử lý dữ liệu đa dạng

Tải từ APIs, xử lý CSV/JSON, tải và làm việc với files dữ liệu, file nén

Agenda

1 Tổng quan về Unix/Linux

Linux và vai trò trong phân tích dữ liệu, bản phân phối phổ biến, triết lý Unix, shell cơ bản, và cấu trúc thư mục

2 Các lệnh cơ bản

Quản lý thư mục/files, xem nội dung, tìm kiếm dữ liệu, vi/vim editor, quản lý processes và phân quyền

3 Xử lý dữ liệu qua command line

Pipe và redirect, filter dữ liệu với grep/awk/sed, xử lý CSV/JSON, xargs cho xử lý song song, tải dữ liệu, làm việc với file nén

Agenda

1 Tổng quan về Unix/Linux

Linux và vai trò trong phân tích dữ liệu, bản phân phối phổ biến, triết lý Unix, shell cơ bản, và cấu trúc thư mục

2 Các lệnh cơ bản

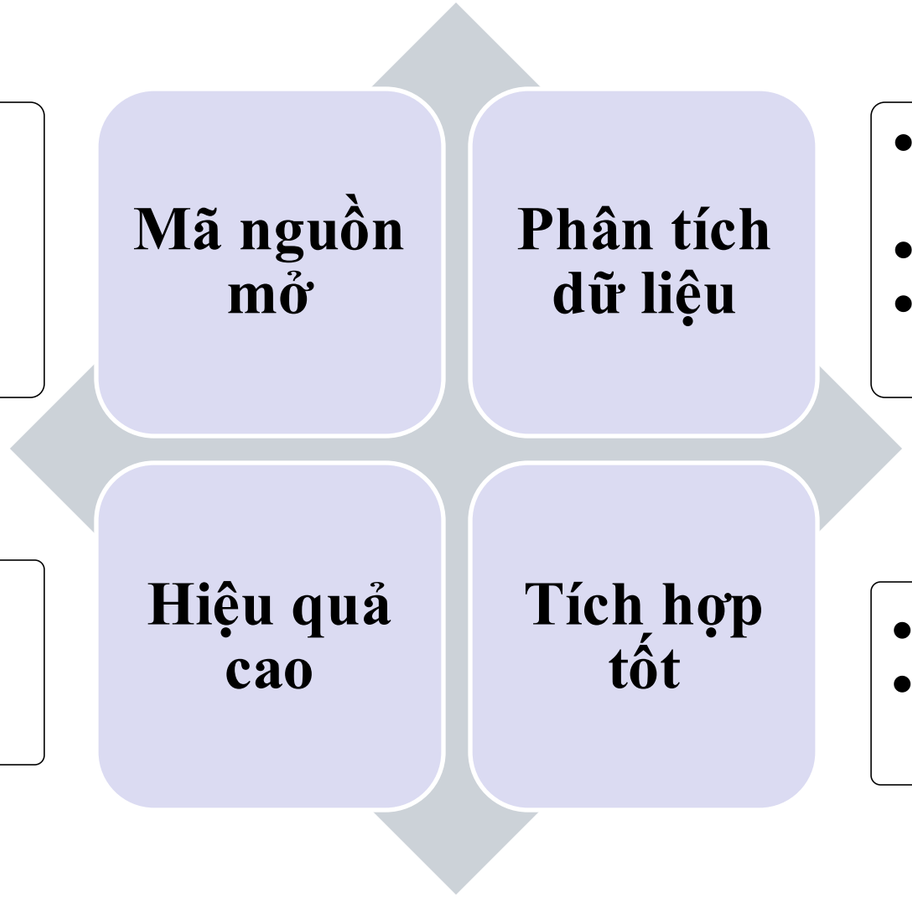
Quản lý thư mục/files, xem nội dung, tìm kiếm dữ liệu, vi/vim editor, quản lý processes và phân quyền

3 Xử lý dữ liệu qua command line

Pipe và redirect, filter dữ liệu với grep/awk/sed, xử lý CSV/JSON, xargs cho xử lý song song, tải dữ liệu, làm việc với file nén

Linux là gì và tại sao cần thiết?

Linux là **hệ điều hành mã nguồn mở phổ biến trong phân tích dữ liệu**, mang lại hiệu quả cao nhờ khả năng xử lý dữ liệu qua command line và tích hợp tốt với công cụ phân tích hiện đại.



- Nền tảng Unix
- Linux Torvalds phát triển (1991)
- Tùy biến hoàn toàn
- Truy cập toàn diện

- Chiếm 90% môi trường high-performance computing
- Nền tảng cho Hadoop, Spark,
- Nền tảng hầu hết các cloud platforms

- Hiệu quả với Data Science
- Cho phép xử lý dữ liệu lớn
- Tự động hóa workflow

- Tích hợp liền mạch với Python/R
- Tích hợp tool phân tích dữ liệu hiện đại

Các bản phân phối Linux phổ biến

Linux có nhiều phiên bản (distributions) đáp ứng nhu cầu đa dạng của Data Scientists: Ubuntu thân thiện với người dùng, Debian ổn định cao, CentOS/RHEL cho doanh nghiệp, và WSL tích hợp với Windows.

Ubuntu

Phổ biến nhất, thân thiện người mới
LTS: hỗ trợ 5 năm, ổn định cho Data Science

Debian

Cực kỳ ổn định, bảo mật cao
Lý tưởng cho nghiên cứu khoa học

CentOS/RHEL

Tiêu chuẩn doanh nghiệp
Hadoop/Big Data, tương thích enterprise

WSL (Windows Subsystem for Linux)

Linux trong Windows
Không cần dual-boot hay máy ảo

Triết lý Unix

Triết lý Unix dựa trên **đơn giản hóa, chuyên biệt hóa và tính nhất quán**, tạo sức mạnh cho xử lý dữ liệu thông qua kết hợp các công cụ nhỏ chuyên biệt.

Nhỏ là đẹp

- Chương trình đơn giản, chuyên biệt
- Công cụ như **grep, sed, awk** giúp xử lý dữ liệu lớn không cần phần mềm phức tạp.

Mỗi chương trình làm một việc thật tốt

- Người phân tích dữ liệu dùng các lệnh riêng biệt như 'sort' để sắp xếp, 'uniq' để lọc, 'cut' để lấy cột từ file CSV

Mọi thứ đều là file

- Dữ liệu, thiết bị, quy trình được xem như file.
- Giúp tương tác qua API, cơ sở dữ liệu, phân vùng thống nhất.

Ưu điểm trong phân tích dữ liệu

- Kết nối lệnh qua pipe (|) tạo luồng xử lý: 'cat data.csv | grep "pattern" | sort | uniq -c > result.txt' để lấy, lọc và đếm dữ liệu chỉ bằng một dòng lệnh.

Shell là gì?

Shell là **giao diện dòng lệnh trung gian giữa người dùng và hệ điều hành**, giúp Data Scientists xử lý dữ liệu hiệu quả qua các lệnh đơn giản.

Giao diện dòng lệnh

- Nơi nhập lệnh điều khiển máy tính.
- Data Scientists dùng shell để xử lý nhanh file dữ nhanh chóng.

Các loại shell phổ biến

- Bash - phổ biến nhất trên Linux/MacOS
- Zsh - cải tiến với tự động hoàn thành
- Fish - dễ dùng, tự gợi ý lệnh từ lịch sử.

Vai trò trung gian

- Chuyển đổi lệnh người dùng thành ngôn ngữ máy tính.
- Giúp xử lý dữ liệu phức tạp mà không cần hiểu sâu về hệ điều hành.

Terminal vs Shell

- Terminal là cửa sổ hiển thị để nhập lệnh. Shell là chương trình xử lý các lệnh đó.
- Ví dụ: iTerm2 (terminal), Bash (shell).

Cấu trúc thư mục trong Linux

Linux tổ chức **hệ thống file theo cấu trúc chuẩn** với các **thư mục chức năng chuyên biệt cho lệnh**, dữ liệu người dùng, cấu hình, dữ liệu biến đổi, ứng dụng và phần mềm bên thứ ba.

Tóm tắt Phần 1

Linux và vai trò với Data Scientists

Hệ điều hành mã nguồn mở cung cấp các công cụ mạnh mẽ để xử lý dữ liệu, tự động hóa quy trình và quản lý tài nguyên hiệu quả.

Các bản phân phối và triết lý Unix

Ubuntu, CentOS, Debian... đều tuân theo triết lý thiết kế đơn giản, tập trung vào chương trình thực hiện một nhiệm vụ cụ thể và hiệu quả.

Shell và Terminal

Terminal là giao diện, Shell là chương trình xử lý lệnh - cung cấp khả năng tương tác hiệu quả với hệ thống.

Cấu trúc thư mục Linux

Hệ thống phân cấp rõ ràng với các thư mục chuyên biệt: /bin, /home, /etc, /var, /usr và /opt - mỗi thư mục đều phục vụ mục đích riêng biệt.

Agenda

1 Tổng quan về Unix/Linux

Linux và vai trò trong phân tích dữ liệu, bản phân phối phổ biến, triết lý Unix, shell cơ bản, và cấu trúc thư mục

2 Các lệnh cơ bản

Quản lý thư mục/files, xem nội dung, tìm kiếm dữ liệu, vi/vim editor, quản lý processes và phân quyền

3 Xử lý dữ liệu qua command line

Pipe và redirect, filter dữ liệu với grep/awk/sed, xử lý CSV/JSON, xargs cho xử lý song song, tải dữ liệu, làm việc với file nén

Các lệnh cơ bản cho Data Science

Các lệnh Unix/Linux thiết yếu giúp Data Scientists **làm việc với thư mục, quản lý files, tìm kiếm và xử lý dữ liệu hiệu quả.**

Lệnh làm việc với thư mục

cd, pwd, ls, mkdir - giúp di chuyển, hiển thị và tạo thư mục mới một cách nhanh chóng

Lệnh quản lý files

touch, cp, mv, rm - tạo, sao chép, di chuyển và xóa files để tổ chức dữ liệu hiệu quả

Lệnh tìm kiếm và xem dữ liệu

cat, head, tail, grep, find - xem nội dung, lọc và tìm kiếm thông tin trong files dữ liệu lớn

Lệnh xử lý dữ liệu

sort, uniq, wc, cut, sed, awk - sắp xếp, loại bỏ trùng lặp, đếm và xử lý dữ liệu theo cột

Lệnh quản lý hệ thống

ps, top, chmod, chown - quản lý processes và phân quyền files khi làm việc với dữ liệu nhạy cảm

Làm việc với thư mục

Các lệnh cơ bản để điều hướng và quản lý thư mục trong Linux, giúp truy cập và tổ chức dữ liệu hiệu quả.

pwd	Hiển thị đường dẫn đầy đủ của thư mục hiện tại, giúp định vị vị trí của bạn trong hệ thống
cd	Di chuyển giữa thư mục. **cd ..** (lên thư mục cha), **cd ~** (về thư mục home)
ls	Liệt kê files và thư mục. **-l** (xem chi tiết), **-h** (kích thước dễ đọc), **-S** (sắp xếp theo kích thước)
ls -la	Hiển thị tất cả files (kể cả ẩn), kèm thông tin quyền, kích thước và thời gian chỉnh sửa
cd ~/datasets	Di chuyển đến thư mục datasets. Ví dụ: **cd ~/projects/covid-analysis/data**

Tạo và quản lý files/thư mục

Các lệnh thiết yếu để tạo, sao chép, di chuyển và xóa files/thư mục trong Linux. Những công cụ này giúp tổ chức dữ liệu hiệu quả.

Tạo, xoá thư mục và files

Tạo thư mục

- **mkdir** datasets (tạo thư mục đơn)
- **mkdir -p** projects/covid_analysis/raw_data (tạo cấu trúc thư mục đa cấp)

Tạo file trống

- **touch** data.csv (tạo file trống)
- **touch** {train,test,validation}.csv (tạo nhiều files cùng lúc)

Xóa

- **rm** temp.csv (xóa file đơn)
- **rm -i** *.tmp (xóa có xác nhận, an toàn với dữ liệu quan trọng)
- **rm -rf** cache/ (cẩn thận! Xóa thư mục và mọi thứ bên trong không thể khôi phục)

Sao chép và di chuyển files/thư mục

Sao chép files

- **cp** source.csv target.csv (sao chép file đơn)
- **cp -r** data_folder backup_folder (sao chép toàn bộ thư mục dữ liệu)

Di chuyển/đổi tên

- **mv** raw_data.csv processed_data.csv (đổi tên file)
- **mv** *.csv ~/projects/analysis/ (di chuyển nhiều files CSV vào thư mục phân tích)

Xem nội dung files

Linux cung cấp nhiều lệnh để xem nội dung files dữ liệu. Lệnh phổ biến: **cat (toàn bộ), less (từng trang), head/tail (đầu/cuối file)** - giúp kiểm tra nhanh cấu trúc và nội dung dữ liệu.

Xem toàn bộ file

cat data.csv

Hiển thị toàn bộ nội dung file. Tránh dùng cho files lớn.

```
1 !cat Linux/data/users_2023-01-01.csv
```

```
user_id,region,status,access_count
user_1_2023-01-01,North,Inactive,28
user_2_2023-01-01,South,Active,42
user_3_2023-01-01,West,Inactive,90
user_4_2023-01-01,North,Inactive,54
```

Xem từng trang

less data.csv

Duyệt qua file lớn. Dùng phím mũi tên để cuộn, phím "/" để tìm kiếm pattern, "q" để thoát.

```
1 !less Linux/data/users_2023-01-01.csv
```

```
user_id,region,status,access_count
user_1_2023-01-01,North,Inactive,28
user_2_2023-01-01,South,Active,42
user_3_2023-01-01,West,Inactive,90
user_4_2023-01-01,North,Inactive,54
```

Xem đầu/cuối file

head -n 10 data.csv

tail -n 20 data.csv

Xem nhanh cấu trúc dữ liệu. **head** để kiểm tra header và định dạng, **tail** để theo dõi log files hoặc dữ liệu mới nhất.

```
1 !head -n 3 Linux/data/users_2023-01-01.csv
```

```
user_id,region,status,access_count
user_1_2023-01-01,North,Inactive,28
user_2_2023-01-01,South,Active,42
```

Tìm kiếm files và dữ liệu

Các lệnh tìm kiếm cơ bản trong Linux giúp định vị và lọc dữ liệu hiệu quả. Bao gồm find để tìm files theo tên/thời gian và grep để tìm nội dung trong files.

find . -name "*.csv"

Tìm tất cả CSV files trong thư mục hiện tại và các thư mục con.

```
1 !find . -name "*.csv"
```

```
./users_2023-01-04.csv
./users_2023-01-01.csv
./users_2023-01-05.csv
./users_2023-01-02.csv
./users_2023-01-03.csv
```

grep "pattern" file.txt

Tìm các dòng chứa mẫu cụ thể trong file. Ví dụ: grep "outlier" results.log để tìm tất cả outliers được ghi trong logs phân tích.

```
1 !grep "precision" model_logs.txt
```

```
2023-01-01 precision: 0.72
2023-01-01 precision: 0.74
2023-01-01 precision: 0.84
2023-01-01 precision: 0.79
2023-01-01 precision: 0.95
```

grep -r "data" .

Tìm nội dung trong tất cả files và thư mục con.
Ví dụ: grep -r "revenue_2023" . để tìm tất cả scripts có sử dụng biến này trong dự án.

find . -name "*.csv" -mtime -7

Tìm files CSV được tạo/sửa trong 7 ngày qua. Ví dụ: find ~/data/raw -name "*.csv" -mtime -1 để kiểm tra datasets mới trong 24 giờ qua.

Thao tác với dữ liệu text cơ bản

Lệnh wc, sort và uniq giúp thao tác với dữ liệu văn bản: đếm dòng/từ, sắp xếp dữ liệu và xác định giá trị trùng lặp.

wc

- Đếm số dòng, từ, ký tự trong file.
- Ví dụ: **wc -l dataset.csv** kiểm tra nhanh số records

```
1 !wc -l model_logs.txt
```

```
100 model_logs.txt
```

sort

- Sắp xếp dữ liệu theo nhiều tiêu chí.
- Ví dụ: **sort -n -k2** results.txt sắp xếp theo cột 2

```
1 !sort -k1 users_2023-01-01.csv
```

```
user_100_2023-01-01,North,Active,76
user_101_2023-01-01,East,Inactive,89
user_10_2023-01-01,East,Inactive,76
user_102_2023-01-01,West,Active,62
user_103_2023-01-01,North,Active,79
```

uniq

- Lọc hoặc đếm dòng trùng lặp (thường dùng sau sort).
- Ví dụ: **sort user_ids.txt | uniq -c** đếm tần suất mỗi user ID,
- **sort categories.txt | uniq -c | sort -nr** tìm danh mục phổ biến nhất.

```
!sort users_2023-01-01.csv | uniq -c
```

```
 1 user_100_2023-01-01,North,Active,76
 1 user_101_2023-01-01,East,Inactive,89
 1 user_10_2023-01-01,East,Inactive,76
 1 user_102_2023-01-01,West,Active,62
 1 user_103_2023-01-01,North,Active,79
```

Chỉnh sửa files với vi/vim

Vi/vim là trình soạn thảo văn bản trong Linux với các chế độ Normal, Insert, Visual, và Command. Công cụ thiết yếu làm việc trên server với tính năng chỉnh sửa, di chuyển và tìm kiếm hiệu quả.

Các chế độ trong vim

- Normal: điều hướng và sử dụng lệnh (mặc định)
- Insert (i, a, o): nhập văn bản
- Visual (v, V, Ctrl+v): chọn văn bản
- Command (:): thực hiện lệnh như lưu, thoát

Lệnh thiết yếu

- :w - lưu file (write)
- :q - thoát (:q! - thoát không lưu)
- :wq hoặc ZZ - lưu và thoát
- u - hoàn tác, Ctrl+r - làm lại
- dd - xóa dòng, yy - sao chép dòng

Di chuyển hiệu quả

- h,j,k,l - trái, xuống, lên, phải
- 0, $ - đầu/cuối dòng
- G - cuối file, gg - đầu file
- /text - tìm kiếm "text"
- :%s/cũ/mới/g - thay thế toàn bộ

Trường hợp sử dụng

- Chỉnh sửa file config.ini cho dự án phân tích
- Sửa đổi script Python trên server không có giao diện
- Xem nhanh CSV headers với :10 (xem 10 dòng đầu)
- Tìm và sửa lỗi trong log files với /error

Biến môi trường

Biến môi trường là các thiết lập hệ thống giúp cấu hình và tối ưu hóa các công cụ phân tích dữ liệu, đường dẫn thư viện, và hiệu suất tính toán cho các tác vụ data science.

PATH	Đường dẫn tìm kiếm các công cụ và lệnh phân tích dữ liệu (Python, R, Julia, TensorFlow, etc.). Thêm với: **export PATH=$PATH:/đường/dẫn/mới**
PYTHONPATH	Chỉ định thư mục chứa modules Python tùy chỉnh cho dự án Data Science. Quan trọng khi làm việc với nhiều dự án phân tích khác nhau.
OPENBLAS_NUM_THREADS	Kiểm soát số threads sử dụng trong các phép tính đại số tuyến tính của NumPy/Pandas. Giúp tối ưu hóa hiệu suất khi xử lý datasets lớn.
JUPYTER_CONFIG_DIR	Đường dẫn đến thư mục cấu hình Jupyter Notebook - công cụ thiết yếu cho Data Scientists.
R_LIBS_USER	Chỉ định thư mục cài đặt packages R tùy chỉnh cho phân tích thống kê.
export VAR=value	Thiết lập biến môi trường tạm thời (session hiện tại). Thêm vào ~/.bashrc để lưu vĩnh viễn.
echo $VAR	Hiển thị giá trị biến. Hữu ích để debug khi scripts ML/DS không tìm thấy thư viện.

Biến môi trường thường được cấu hình trong file **~/.bashrc** hoặc **~/.bash_profile** để tự động áp dụng khi đăng nhập vào hệ thống.

Quyền và phân quyền files

Linux sử dụng **hệ thống phân quyền rwx (read-write-execute) cho từng đối tượng (user-group-others)** để kiểm soát truy cập files. Permissions đóng vai trò quan trọng trong bảo mật và quản lý dữ liệu khoa học.

Cấu trúc quyền trong Linux	
rwx	read (đọc dữ liệu), write (ghi/sửa dữ liệu), execute (chạy scripts)
u	user (người sở hữu)
g	group (nhóm)
o	others (người dùng khác)
rw-r--r--	File dữ liệu chỉ có thể đọc với group và others

Lệnh phân quyền	
chmod 755	Cho phép chạy script Python xử lý dữ liệu
chmod u+x	Thêm quyền thực thi cho script huấn luyện model
chown	Chuyển quyền sở hữu dataset
chmod 400	Bảo vệ file chứa API keys chỉ để đọc

Quản lý processes cho dự án Data Science

Quản lý processes giúp theo dõi, kiểm soát và tối ưu hóa các tác vụ phân tích dữ liệu và huấn luyện model trong dự án Data Science.

ps aux | grep python

Liệt kê tất cả các processes Python đang chạy, giúp theo dõi các script phân tích dữ liệu và model training đang hoạt động

top -u username

Hiển thị processes theo thời gian thực của user cụ thể, hữu ích khi theo dõi tài nguyên CPU/RAM cho các tác vụ ML nặng

kill -9 PID

Dừng process theo ID khi model training bị treo hoặc script xử lý dữ liệu chiếm quá nhiều tài nguyên

nohup python train.py &

Chạy script huấn luyện model ở background và duy trì chạy kể cả khi đăng xuất khỏi server, kết quả được lưu vào nohup.out

Bài tập thực hành 1

Mục tiêu: Thực hành các lệnh cơ bản để thiết lập dự án phân tích dữ liệu, bao gồm tạo cấu trúc thư mục, cấu hình quyền truy cập, tìm kiếm files và kiểm tra cấu trúc hệ thống.

1. Tạo cấu trúc thư mục dự án

Tạo một cấu trúc thư mục đã định sẵn cho dự án bao gồm các thư mục con **data**, **scripts**, và **results** bên trong một thư mục chính tên là **project**

2. Thiết lập quyền bảo mật

Thay đổi quyền truy cập cho tất cả các tệp trong thư mục **project/scripts/** thành **750** (cho phép chủ sở hữu đọc, ghi, thực thi; nhóm đọc, thực thi; những người khác không có quyền)

3. Tìm tất cả file CSV

Tìm và liệt kê tất cả các tệp có đuôi .csv bên trong thư mục project và các thư mục con của nó.

4. Kiểm tra cấu trúc và quyền truy cập

Xác minh lại cấu trúc thư mục và quyền truy cập của các tệp và thư mục trong project

Bài tập thực hành 1 (Gợi ý)

1. Tạo cấu trúc thư mục dự án

Sử dụng lệnh mkdir -p để tạo thư mục cha và các thư mục con cùng một lúc.

```
project
├── data
├── results
└── scripts

3 directories, 0 files
```

2. Thiết lập quyền bảo mật

Sử dụng lệnh chmod -R để thay đổi quyền truy cập cho thư mục và các tệp/thư mục con bên trong nó một cách đệ quy.

```
total 20
drwxr-xr-x 5 root root 4096 Jun  5 01:53 .
drwxr-xr-x 1 root root 4096 Jun  5 01:53 ..
drwxr-xr-x 2 root root 4096 Jun  5 01:53 data
drwxr-xr-x 2 root root 4096 Jun  5 01:53 results
drwxr-xr-x 2 root root 4096 Jun  5 01:53 scripts
```

```
total 20
drwxr-xr-x 5 root root 4096 Jun  5 01:53 .
drwxr-xr-x 1 root root 4096 Jun  5 01:53 ..
drwxr-xr-x 2 root root 4096 Jun  5 01:53 data
drwxr-xr-x 2 root root 4096 Jun  5 01:53 results
drwxr-x--- 2 root root 4096 Jun  5 01:53 scripts
```

Bài tập thực hành 1 (Gợi ý)

3. Tìm tất cả file CSV

Sử dụng lệnh find với tùy chọn -name để tìm theo tên và -type f để chỉ tìm tệp.

```
!find project -name "*.csv" -type f
```

```
project/data/users_2023-01-01.csv
```

4. Kiểm tra cấu trúc và quyền truy cập

Sử dụng lệnh ls -la để hiển thị thông tin chi tiết, bao gồm quyền truy cập.

```
total 20
drwxr-xr-x 5 root root 4096 Jun  5 01:53 .
drwxr-xr-x 1 root root 4096 Jun  5 01:53 ..
drwxr-xr-x 2 root root 4096 Jun  5 01:53 data
drwxr-xr-x 2 root root 4096 Jun  5 01:53 results
drwxr-x--- 2 root root 4096 Jun  5 01:53 scripts
```

Tóm tắt Phần 2

Làm việc với files & thư mục

Tạo, di chuyển, sao chép và xóa files/thư mục; xem nội dung files; tìm kiếm dữ liệu

Thao tác text & chỉnh sửa

Xử lý dữ liệu text cơ bản; chỉnh sửa files với vi/vim; làm việc với biến môi trường

Quyền truy cập & quản lý

Phân quyền files cho dự án; quản lý processes cho các tác vụ ML & data processing

Bài tập thực hành

Xây dựng cấu trúc thư mục dự án; cài đặt quyền bảo mật; tìm kiếm files; kiểm tra cấu trúc

Agenda

1 **Tổng quan về Unix/Linux**

Linux và vai trò trong phân tích dữ liệu, bản phân phối phổ biến, triết lý Unix, shell cơ bản, và cấu trúc thư mục

2 **Các lệnh cơ bản**

Quản lý thư mục/files, xem nội dung, tìm kiếm dữ liệu, vi/vim editor, quản lý processes và phân quyền

3 **Xử lý dữ liệu qua command line**

Pipe và redirect, filter dữ liệu với grep/awk/sed, xử lý CSV/JSON, xargs cho xử lý song song

Pipe và Redirect - công cụ mạnh mẽ

Pipe và Redirect là các công cụ thiết yếu trong Linux để **điều hướng luồng dữ liệu, kết nối các lệnh và lưu trữ kết quả**. Chúng cho phép xây dựng các pipeline xử lý dữ liệu phức tạp, hiệu quả cho công việc.

- Ba luồng dữ liệu chuẩn trong Linux
- stdin nhận input
- stdout xuất kết quả
- stderr hiển thị lỗi

stdin
stdout
stderr

Pipe (|)

- Kết nối output của lệnh này với input của lệnh khác
- Xây dựng pipeline cho dữ liệu phức tạp
- cat data.csv | grep "2023" | sort

Input (<)

- Lấy input từ file thay vì bàn phím
- Tự động hóa các tác vụ với dữ liệu lớn
- python process.py < config_parameters.txt

Redirect (>, >>)

- Chuyển output ra file
- > ghi đè, >> thêm vào
- python train.py > training_log.txt 2> errors.log

Filter commands cho Data

Các lệnh filter trong Linux (sed, awk, cut, tr, grep) là công cụ mạnh mẽ giúp xử lý, biến đổi và trích xuất thông tin từ dữ liệu dạng văn bản, đặc biệt hiệu quả khi kết hợp với pipe và redirect.

sed	sed 's/old/new/g' data.csv sed -n '1,5p' large_dataset.txt	Thay thế văn bản, xử lý regex và lọc dòng Ví dụ: thay thế định dạng ngày hoặc rút gọn tập dữ liệu
awk	awk -F',' '{sum+=$3} END {print "Avg:",sum/NR}' metrics.csv	Xử lý theo cột với ngôn ngữ lập trình hoàn chỉnh Mạnh mẽ cho tính toán thống kê, tổng hợp và biến đổi dữ liệu
cut	cut -d',' -f1,3-5 sales_data.csv cut -c1-10 log_file.txt	Trích xuất cột hoặc ký tự từ dữ liệu có cấu trúc Lý tưởng cho việc loại bỏ cột không cần thiết trước khi phân tích
tr	tr ',' '\t' < data.csv > data.tsv cat file.txt | tr -d '\r' | tr '[:upper:]' '[:lower:]'	Thay thế hoặc xóa ký tự, chuyển đổi định dạng file Hữu ích để chuẩn hóa dữ liệu trước khi đưa vào mô hình
grep	grep -i "error" log_files/*.log grep -A 3 "Exception" app.log	Tìm kiếm chuỗi/pattern trong nhiều files Rất hiệu quả khi phân tích logs và debug các ứng dụng ML

Trích xuất dữ liệu từ files lớn

Linux cung cấp nhiều công cụ để xử lý files dữ liệu lớn mà không cần tải toàn bộ vào bộ nhớ. Bạn có thể xem từng phần, chia nhỏ file và áp dụng kỹ thuật xử lý hiệu suất cao cho dữ liệu lớn.

Xem một phần của file lớn

head -n 1000 bigfile.csv | less (xem 1000 dòng đầu)

tail -n 500 bigfile.csv > sample.csv (trích xuất 500 dòng cuối)

sed -n '1000,2000p' bigfile.csv (xem dữ liệu từ dòng 1000-2000)

Chia nhỏ files để xử lý hiệu quả

split -l 10000 bigfile.csv part_ (chia theo số dòng)

split -b 100M bigfile.csv part_ (chia theo kích thước)

split -n l/10 bigfile.csv chunk_ (chia thành 10 phần bằng nhau)

Xử lý từng phần với hiệu suất cao

awk 'NR % 100000 == 0 {print NR/1000000 " million lines processed"}' bigfile.csv

find ./chunks -name "part_*" | xargs -P 4 -I{} sh -c 'grep "pattern" {} > {}.results'

Kết hợp kỹ thuật streaming + pipe để giảm sử dụng RAM

Xử lý dữ liệu JSON với jq (1/2)

jq là công cụ command-line để trích xuất, lọc và biến đổi dữ liệu JSON. Cho phép data scientists thực hiện các thao tác phức tạp trên dữ liệu JSON mà không cần code phức tạp.

Cài đặt jq

apt-get install jq # Ubuntu/Debian#

brew install jq # macOS

```
1 !apt-get install jq  # Ubuntu/Debian
```

```
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libjq1 libonig5
The following NEW packages will be installed:
  jq libjq1 libonig5
```

Trích xuất đối tượng từ mảng results

cat data.json | jq '.results[]'

```
1 !cat data.json | jq '.results[]'
```

```
{
  "name": "Model_A",
  "score": 0.85,
  "type": "numeric",
  "value": 42.5,
  "accuracy": 0.92,
  "created_at": "2024-01-15"
}
```

Lọc dữ liệu theo điều kiện

cat data.json | jq '.results[] | select(.score > 0.8)'

```
1 !cat data.json | jq '.results[] | select(.score > 0.8)'
```

```
{
  "name": "Model_A",
  "score": 0.85,
  "type": "numeric",
  "value": 42.5,
  "accuracy": 0.92,
  "created_at": "2024-01-15"
}
```

Xử lý dữ liệu JSON với jq (2/2)

jq là công cụ command-line để trích xuất, lọc và biến đổi dữ liệu JSON. Cho phép data scientists thực hiện các thao tác phức tạp trên dữ liệu JSON mà không cần code phức tạp.

Biến đổi dữ liệu - tạo cấu trúc JSON mới cho phân tích

cat data.json | jq '.results[] | {name, value: .score}'

```
!cat data.json | jq '.results[] | {name, value: .score}'
```

```
{
  "name": "Model_A",
  "value": 0.85
}
{
  "name": "Model_B",
  "value": 0.73
}
```

Tổng hợp dữ liệu - thống kê cơ bản

cat data.json | jq '.results | length' # Đếm số kết quảcat data.json | jq '[.results[].score] | add/length' # Điểm trung bình

```
!cat data.json | jq '[.results[].score] | add/length'
```

```
0.805
```

Kết hợp với pipeline để tiền xử lý

cat data.json | jq '.results[] | select(.type=="numeric") | .value' > numbers.txt

```
!cat data.json | jq '.results[] | select(.type=="numeric") | .value' > numbers.txt
!ls
```

```
data.json  numbers.txt  sample_data
```

numbers.txt

```
42.5
38.2
31.9
47.3
```

Xử lý dữ liệu CSV qua command line

Command line cung cấp công cụ mạnh mẽ để đọc, lọc và phân tích dữ liệu CSV mà không cần phần mềm chuyên dụng. Các lệnh như grep, cut, awk và sort giúp xử lý hiệu quả với datasets lớn.

Đọc và lọc dữ liệu

grep "POSITIVE" sentiment_analysis.csv | cut -d',' -f1,3

Trích xuất cột 1 (ID) và cột 3 (điểm dự đoán) từ các dòng có chứa "POSITIVE" trong file phân tích cảm xúc

Chuyển đổi định dạng

csvkit: csvjson customer_data.csv > customer_data.json

Chuyển đổi dataset khách hàng từ CSV sang JSON để tích hợp vào ứng dụng web hoặc API

Tính toán thống kê

awk -F',' '{sum+=$5; count++} END {print "Giá trị trung bình cột 5 (giá): " sum/count}' sales.csv

Nối lệnh với head/tail để phân tích nhanh khi làm việc với bộ dữ liệu hàng triệu dòng

Sắp xếp và lọc dữ liệu

sort -t',' -k3,3nr product_metrics.csv | head -n 10 > top10_products.csv

Sắp xếp dataset theo cột 3 (doanh số) giảm dần và xuất 10 sản phẩm hàng đầu ra file mới

Phân tích logs với command line

Command line cung cấp công cụ mạnh mẽ để trích xuất, lọc và phân tích logs hệ thống. Các lệnh cơ bản như grep, awk và sort giúp xác định lỗi, phát hiện xu hướng và theo dõi hiệu suất hệ thống.

Tìm và phân loại lỗi

```
grep "ERROR" app.log | grep -v "Connection timeout" | tail -n 50
```

- Lọc 50 lỗi gần nhất từ app.log
- Loại trừ các timeout.
- Hữu ích khi debug mô hình ML trong production.

Đếm và phân tích xu hướng

```
grep "prediction_failed" model.log | awk '{print $1}' | cut -d'-' -f1 | sort | uniq -c
```

- Thống kê số lượng dự đoán thất bại theo ngày
- Giúp xác định xu hướng suy giảm hiệu suất mô hình theo thời gian.

Thống kê mã phản hồi API

```
awk '{print $9}' api_access.log | sort | uniq -c | sort -nr | head -10
```

- Hiển thị 10 mã phản hồi phổ biến nhất từ log API
- Giúp phát hiện vấn đề khi thu thập dữ liệu qua API cho các dự án phân tích.

Kết hợp các commands phức tạp

Pipeline là cách kết hợp nhiều lệnh đơn giản thành một dãy xử lý mạnh mẽ. Sử dụng pipe (|) để kết nối output của lệnh trước với input của lệnh sau, giúp phân tích dữ liệu phức tạp trong một dòng lệnh duy nhất.

Phân tích top users từ logs

```
cat access.log |  awk '{print $1}' |  sort |  uniq -c |  sort -nr |  head -10
```

```
 7 192.168.1.100
 6 10.0.0.50
 5 203.0.113.25
 4 198.51.100.75
 3 172.16.0.10
```

Sử dụng tee để ghi và hiển thị

```
cat data.csv |  grep "2023" |  cut -d',' -f2,3 |  tee filtered.csv |  head
```

```
Tran Thi B,Marketing
Le Van C,Finance
Hoang Van E,IT
Vu Thi F,Marketing
Bui Thi H,IT
Ngo Van I,HR
Mai Van L,Finance
Cao Thi M,IT
```

xargs - xử lý dữ liệu song song

Công cụ mạnh mẽ chuyển đổi input thành arguments cho lệnh khác, hỗ trợ xử lý dữ liệu song song và tối ưu hiệu suất cho các tác vụ phân tích dữ liệu lớn.

Chuyển input thành arguments

find datasets/ -name ".csv" | xargs wc -l*

Đếm số dòng của tất cả file CSV trong thư mục datasets, giúp xác định kích thước dữ liệu trước khi phân tích

Parameter substitution

find data/ -name "sensor_*.txt" | xargs -I {} bash -c 'echo "Processing {}"; cat {} | cut -d"," -f1,3 > {}.processed'

Xử lý từng file sensor data, trích xuất cột 1 và 3 thành file mới

Xử lý song song

ls model_*.csv | xargs -P 8 -I {} python preprocess.py --input {} --output {}.clean

Tiền xử lý 8 file dữ liệu mô hình cùng lúc, tận dụng nhiều cores CPU

Tăng hiệu suất phân tích

find logs/ -name ".json" | xargs -P 4 -I {} jq '.results[]' {} > combined_results.json*

Xử lý song song nhiều file logs định dạng JSON, kết hợp kết quả phân tích vào một file để visualize

<Case study>

Xử lý dữ liệu thực tế

Case study: Kết hợp lệnh Unix để phân tích log máy chủ và tạo báo cáo tự động, từ dữ liệu thô đến báo cáo cuối cùng.

Phân tích log máy chủ

```
!cat access.log | grep "POST /api" | awk '{print $4, $5, $7}' | sort | uniq -c | sort -nr > api_usage.txt
```

Tạo báo cáo tự động

```
echo "API Usage Report $(date)" > report.txt
echo "-----------------------" >> report.txt
cat api_usage.txt | head -10 >> report.txt
```

api_usage.txt

```
      1 [15/Jan/2024:10:47:12 +0000] /api/submit
      1 [15/Jan/2024:10:35:23 +0000] /api/update
      1 [15/Jan/2024:10:30:23 +0000] /api/data
      1 [15/Jan/2024:10:24:12 +0000] /api/login

```

report.txt

```
API Usage Report Sat Jun  7 12:56:03 AM UTC 2025
-----------------------
      1 [15/Jan/2024:10:47:12 +0000] /api/submit
      1 [15/Jan/2024:10:35:23 +0000] /api/update
      1 [15/Jan/2024:10:30:23 +0000] /api/data
      1 [15/Jan/2024:10:24:12 +0000] /api/login

```

Tải dữ liệu từ Internet (wget, curl)

Hai công cụ command line phổ biến để tải dữ liệu: wget phù hợp với dữ liệu lớn, khả năng tiếp tục tải khi bị ngắt; curl linh hoạt hơn cho tương tác với APIs và xác thực.

wget		
wget	url	tải dataset từ nguồn mở
wget -O boston_housing.csv	url	lấy và save vào file
wget -c	url	tiếp tục tải khi bị ngắt kết nối
wget -I	list_urls	tải nhiều files từ list URLs

curl		
curl	url	Lấy thông tin từ url
curl -o stock_data.json	url	đổi tên file
curl -u username:password	url	truy cập API yêu cầu xác thực
curl -H "Authorization: Bearer token "	list_urls	tải nhiều files từ list URLs

Làm việc với files nén

Các lệnh **tar và gzip giúp đóng gói, nén và giải nén dữ liệu**, tiết kiệm không gian lưu trữ và băng thông khi chia sẻ. Đặc biệt hữu ích khi làm việc với datasets lớn, cho phép xử lý dữ liệu nén mà không cần giải nén hoàn toàn.

tar	tar -cvf dataset.tar raw_data/	Đóng gói thư mục dữ liệu thô mà không nén, hữu ích khi cần lưu trữ nhiều file CSV liên quan
tar + gzip	tar -czvf project_data.tar.gz datasets/	Đóng gói và nén dữ liệu project, giảm kích thước 60-70%, thích hợp khi chia sẻ dữ liệu qua email hoặc lưu trữ
Giải nén	tar -xzvf kaggle_dataset.tar.gz -C ~/projects/	Giải nén dataset tải về từ Kaggle vào thư mục project cụ thể
Xem nội dung	tar -tvf model_backups.tar	Kiểm tra các phiên bản model đã lưu trữ mà không cần giải nén toàn bộ
Xử lý không giải nén	zcat large_logs.gz | grep "ERROR" | head -n 20	Phân tích lỗi trong file logs lớn đã nén mà không cần giải nén toàn bộ, tiết kiệm không gian đĩa
Xử lý nhiều file	find . -name "*.gz" -exec zcat {} \; | awk '{sum += $1} END {print sum}'	Tính tổng giá trị từ nhiều file dữ liệu nén mà không cần giải nén riêng lẻ

Giải nén và xử lý nhiều files

AI VIETNAM
@aivietnam.edu.vn

Giải nén hàng loạt

```
for file in *.tar.gz;
  do  tar -xzvf "$file“
done
```

Giải thích: Duyệt qua tất cả file .tar.gz trong thư mục hiện tại và giải nén từng file một cách tự động.

Xử lý dữ liệu từ nhiều files

```
for file in *.gz; do
 zcat "$file" | grep "ERROR" >>
all_errors.txt
 done
```

Giải thích: Đọc nội dung từ tất cả file .gz, tìm các dòng chứa từ "ERROR" và ghi tất cả vào file all_errors.txt

Xử lý báo cáo hàng ngày

```
for report in report_*.gz; do
   date=$(echo $report | sed
's/report_\(.*\)\.gz/\1/')
   zcat "$report" | awk '{sum+=$2} END {print
"""$date""", sum}'
done > summary.csv
```

Giải thích: Xử lý các file báo cáo theo ngày, trích xuất ngày từ tên file, tính tổng cột thứ 2 trong mỗi file và xuất kết quả ra file CSV với format: ngày, tổng.

Bài tập thực hành 2

Mục tiêu: tập trung vào kỹ năng xử lý dữ liệu thực tế với command line: phân tích CSV, xử lý JSON, phân tích logs và tự động hóa quy trình báo cáo.

1. Xử lý file CSV người dùng

Xử lý users_2023-01-01.csv: Đếm người dùng theo khu vực. Lọc người dùng theo "Active". Sắp xếp theo tần suất truy cập (giảm dần).

2. Phân tích dữ liệu JSON thời tiết

Phân tích file weather_hanoi.json. Trích xuất nhiệt độ, độ ẩm theo ngày. Tính nhiệt độ & độ ẩm trung bình.

3. Tạo báo cáo hiệu suất mô hình

Phân tích file model_logs.txt để tổng hợp precision, recall theo ngày và phát hiện xu hướng suy giảm (Đếm precision < 0.75 và recall < 0.75 theo ngày)

4. Tự động hóa quy trình báo cáo

- Tạo thư mục project/reports.
- Kết hợp lệnh, lưu kết quả vào project/reports/weekly.txt.
- Tạo file data (region_counts.dat) cho biểu đồ.
- Dùng gnuplot vẽ biểu đồ người dùng theo khu vực (users_by_region.png)

Bài tập thực hành 2 (Gợi ý)

AI VIETNAM
@aivietnam.edu.vn

1. Xử lý file CSV người dùng

- Đếm: cut ... | tail ... | sort | uniq –c
- Lọc: grep "Active" ...
- Sắp xếp: sort -t',' -k4,4nr ...

user_id	region	status	access_count
user_1_2023-01-01	North	Inactive	28
user_2_2023-01-01	South	Active	42
user_3_2023-01-01	West	Inactive	90
user_4_2023-01-01	North	Inactive	54
user_5_2023-01-01	East	Active	18
user_6_2023-01-01	South	Inactive	31

Đếm

```
51 East
61 North
40 South
48 West
```

Lọc

```
user_86_2023-01-01,North,Active,93
user_87_2023-01-01,North,Active,65
user_91_2023-01-01,West,Active,24
user_92_2023-01-01,South,Active,86
user_94_2023-01-01,East,Active,27
user_97_2023-01-01,South,Active,93
user_98_2023-01-01,East,Active,77
```

Sắp xếp

```
user_111_2023-01-01,North,Inactive,100
user_122_2023-01-01,South,Active,100
user_53_2023-01-01,South,Active,100
user_60_2023-01-01,West,Active,100
user_7_2023-01-01,North,Inactive,100
user_123_2023-01-01,South,Active,99
user_148_2023-01-01,West,Inactive,99
```

Bài tập thực hành 2 (Gợi ý)

2. Phân tích dữ liệu JSON thời tiết

- Trích xuất: cat ... | jq '.days[] | .current | [...]
- Trung bình: cat ... | jq '[...] | add/length'

```
[
  20.5,
  77
]
```

```
[
  20.7,
  77.2
]
```

```
[
  19.3,
  57.8
]
```

3. Tạo báo cáo hiệu suất mô hình

Đếm (precision/recall): grep "..." ... | awk '$3 < 0.75 {print $1}' | cut ... | sort | uniq -c

Precision

```
  2 2023-01-01
  2 2023-01-03
  3 2023-01-04
  3 2023-01-05
  2 2023-01-06
  2 2023-01-07
  2 2023-01-09
  2 2023-01-10
```

Recall

```
5 2023-01-01
2 2023-01-02
2 2023-01-03
2 2023-01-04
3 2023-01-05
4 2023-01-06
4 2023-01-07
3 2023-01-08
4 2023-01-09
4 2023-01-10
```

Bài tập thực hành 2 (Gợi ý)

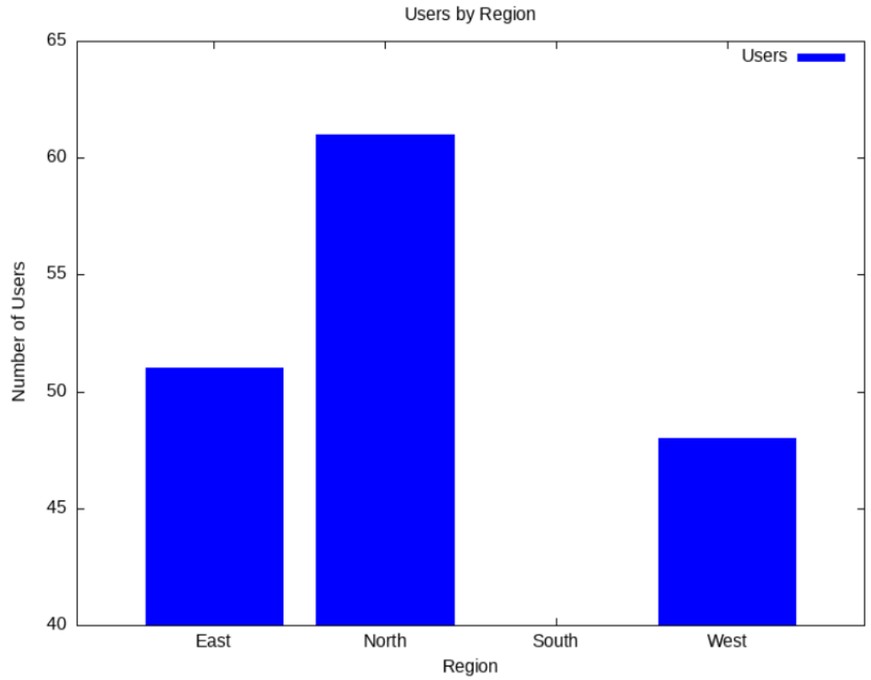
AI VIETNAM
@aivietnam.edu.vn

4. Tự động hóa quy trình báo cáo

- Tạo thư mục: mkdir -p project/reports
- Ghi file: echo "..." >> weekly.txt, ... | tee region_counts.dat
- Biểu đồ: gnuplot create_charts.gp (Nội dung create_charts.gp gồm: set terminal png ..., set output ..., plot 'project/reports/region_counts.dat' ...)

```
START OF REPORT
1. USERS BY REGION
==================
    51 East
    61 North
    40 South
    48 West
2. WEATHER ANALYSIS
===========================
Average Temperature:
24.089999999999996
Average Humidity:
66.42333333333333
```

```
3. MODEL PERFORMANCE
============================
Counts Precision < 0.75:
    2 2023-01-01
    2 2023-01-03
    3 2023-01-04
    3 2023-01-05
    2 2023-01-06
    2 2023-01-07
    2 2023-01-09
    2 2023-01-10
END OF REPORT
```



Tóm tắt Phần 3

Pipe và Redirect

Kết nối đầu ra của lệnh này với đầu vào của lệnh khác, tạo thành pipeline xử lý dữ liệu hiệu quả

Filter commands và trích xuất dữ liệu

Sử dụng grep, sed, awk để lọc, tìm kiếm và trích xuất thông tin từ các tập tin lớn

Xử lý dữ liệu có cấu trúc

Thao tác với dữ liệu JSON (jq), CSV và phân tích logs thông qua command line

Kết hợp commands phức tạp

Xây dựng các pipeline phức tạp để xử lý và phân tích dữ liệu từ nhiều nguồn

Xử lý song song với xargs

Tận dụng xargs để chạy nhiều tác vụ cùng lúc, tối ưu hiệu suất cho phân tích dữ liệu lớn

Tổng kết

Nền tảng Unix/Linux vững chắc

Hiểu biết về triết lý Unix, cấu trúc hệ thống và shell giúp Data Scientists làm việc hiệu quả hơn trong môi trường phân tích dữ liệu phức tạp.

Xử lý dữ liệu qua command line

Các lệnh Unix/Linux cung cấp công cụ mạnh mẽ để xử lý, lọc và phân tích dữ liệu lớn nhanh chóng mà không cần load toàn bộ vào bộ nhớ.

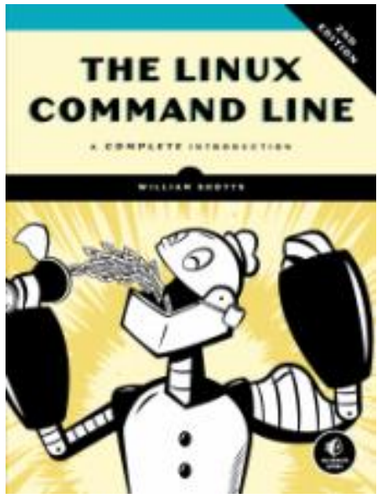
Kỹ năng quản lý và xử lý dữ liệu

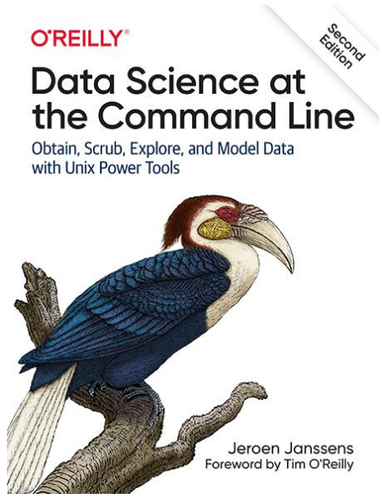
Làm chủ các công cụ như pipe, redirect, jq và xargs giúp Data Scientists độc lập hơn trong việc quản lý dữ liệu từ nhiều nguồn khác nhau.

Tài liệu tham khảo

Sách

- **The Linux Command Line** - William Shotts (Nền tảng Unix/Linux toàn diện)
- **Data Science at the Command Line** - Jeroen Janssens (Tập trung vào xử lý dữ liệu)





Online

- **Linux Journey** (linuxjourney.com) - Học tương tác miễn phí từ cơ bản đến nâng cao
- **Codecademy Bash/Shell** - Khóa học thực hành với bài tập ETL dữ liệu
- **DataCamp Command Line** - Series "Bash for Data Science" với 24+ bài tập thực tế