DSI Notes

Kevin Juandi
email: kjuandi@gmail.com

April 5, 2022

Contents

Chapter 1

Setting things up

1.1 Installing Python and Jupyter Notebook

This section is about how to install python and jupyter notebook. I will update it later. Hopefully will be easily followed.

1.2 The Command Line

In the past, computers didn't have graphical user interface (GUI, pronounced "gooey"). Instead, everyone interacted with the computer using text commands in what we call a command-line interface (CLI). DOS and MS-DOS are perhaps the most well known command line interface operating system. Some Linux distribution still lack GUI and operated text based.

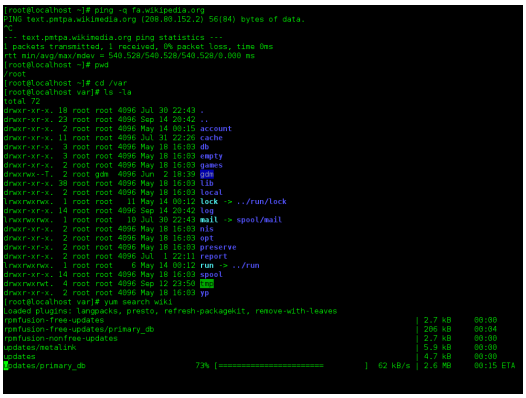


Figure 1.1: Linux Command Line.

A shell (or terminal) is a type of command-line program that contains a simple, text-based user interface, enabling us to access all of an operating

system's services. It is, put simply, a program that accepts text as an input and translates that text into the appropriate functions you want your computer to run.
It might look cumbersome but everything we can do with GUI can also be done with command line, often faster. It just doesn't look pretty.

1.2.1 How to Access Terminal

It is extremely simple for Linux users, you can just type "terminal" on your program search.
Windows users have several options:

- **Windows Command Prompt**: Also known as cmd. A legacy DOS-based shell
- **Windows PowerShell**: The official Windows-native shell and scripting language, intended to replace the antiquated Command Prompt.

Alternatively, there are several others third party programs available, such as GitBash and the built in cmd-prompt and PowerShell in Anaconda.

1.2.2 Shell Commands

Here is a list of commonly used shell commands.

Commands	
ls	list directory
cd	change directory
cat	read, create, concatenate files
mv	move or rename directory
echo	print text to terminal window
touch	create files
mkdir	make directory
man	print manual or get help
pwd	print working directory
rmdir	remove directory
cp	copy data or file
head	read the start of file
tail	read the end of file
exit	exit out of a directory
kill	terminate a process

1.2.3 Navigating with Terminal

1.2.4 Shell Demonstrations

image intensive

1.3 GIT

ALWAYS DOUBLE CHECK BEFORE MERGE. Reverting a merge can be a real pain.

Chapter 2

Python Variables and Data Types

ref: python for everybody

Chapter 3

Control Flow

3.1 Conditional Statement

3.2 Loop

Chapter 4

List and Dictionary

Chapter 5

Numpy and Matplotlib

5.1 Linear Algebra

This section would only have very basic of linear algebra, adapted from Artin's algebra textbook[1]. This is not intended for mathematics course which is why we would keep it to bare minimum
Let m and n be positive integers. An $m \times n$ matrix is a collection of mn numbers arranged in rectangular array.

$$\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{m1} & \cdots & a_{mn} \end{bmatrix}$$

Figure 5.1: $m \times n$ matrix.

For example, $\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix}$ is a 2×3 matrix.
Numbers in a matrix are the *matrix entries*. They are usually denoted as a_{ij} where i and j are indices with $1 \leq i \leq m$ and$1 \leq j \leq n$.
An $n \times n$ matrix is called *square matrix*. An $1 \times n$ matrix is an n-dimensional row vector.
Let $A = (a_{ij})$ and $B = (b_{ij})$ be two $m \times n$ matrix. Their sum $A + B$ is the $m \times n$ matrix $S = (s_{ij})$ defined by:

$$s_{ij} = a_{ij} + b_{ij} \tag{5.1}$$

Thus

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 3 \\ 4 & -3 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 & 3 \\ 5 & 0 & 6 \end{bmatrix} \tag{5.2}$$

Scalar multiplication of an $m \times n$ matrix A by a number c is another $m \times n$ matrix $B = (b_{ij})$, where $(b_{ij} = ca_{ij}$ for all i, j. Thus

$$2\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix} = \begin{bmatrix} 4 & 2 & 0 \\ 2 & 6 & 10 \end{bmatrix} \tag{5.3}$$

Things start to get nasty when we come to Matrix multiplication. Before we get there, it's imperative to first learn about the most basic form of matrix, vectors.
Let A be a row vector and B a column vector of the same size, let say m. If the entries of A and B are denoted by a_i and b_i respectively, the (dot) product of AB is a 1×1 matrix or scalar.

$$\begin{bmatrix} a_1 & a_2 & \cdots & a_m \end{bmatrix} \times \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix} = a_1b_1 + a_2b_2 + \cdots + a_mb_m \tag{5.4}$$

Thus

$$\begin{bmatrix} 1 & 3 & 5 \end{bmatrix} \times \begin{bmatrix} 1 \\ -1 \\ 4 \end{bmatrix} = 1 - 3 + 20 = 18 \tag{5.5}$$

The entries of product matrix are computed by multiplying all rows of A by all columns of B. If we denote the product matrix AB by $P = (p_{ij})$, then

$$p_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + \cdots + a_{im}b_{mj} \tag{5.6}$$

For example,

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \\ 4 \end{bmatrix} = \begin{bmatrix} 1 \\ 18 \end{bmatrix} \tag{5.7}$$

Care should be taken that the product of matrix multiplication is non-commutative, $AB \not\equiv BA$.
Transpose of a matrix A is a matrix A^T with it's row and column flipped.

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix}^T = \begin{bmatrix} 2 & 1 \\ 1 & 3 \\ 0 & 5 \end{bmatrix} \tag{5.8}$$

Matrix A is called invertible if there is $n \times n$ square matrix B such that $AB = BA = I_n$ where I_n is $n \times n$ identity matrix

5.2 Numpy

Numpy is a python library for handling large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

5.3 Matplotlib

will be image intensive, set a folder for that

Chapter 6

Pandas

6.1 Introduction to Pandas

Pandas is a popular python library for dealing with database. It is one of our primary tool in this course. Let us first import Pandas.

```
import pandas as pd
import numpy as np
```

Figure 6.1: Importing Pandas as pd

We would then load our dataset and import them as Pandas dataset.

```
drug = pd.read_csv('../../../../resource-datasets/drug_use_by_age/drug-use-by-age.csv')
```

Figure 6.2: Importing data

We can simply call the data by typing it's name (in this case *drug*).

```
drug
```

	age	n	alcohol-use	alcohol-frequency	marijuana-use	marijuana-frequency	cocaine-use	cocaine-frequency	crack-use	crack-frequency	...	oxycontin-use	oxycontin-frequency	tranquilizer-use	tranquilizer-frequency	stimula
0	12	2798	3.9	3.0	1.1	4.0	0.1	5.0	0.0	-	...	0.1	24.5	0.2	52.0	
1	13	2757	8.5	6.0	3.4	15.0	0.1	1.0	0.0	3.0	...	0.1	41.0	0.3	25.5	
2	14	2792	18.1	5.0	8.7	24.0	0.1	5.5	0.0	-	...	0.4	4.5	0.9	5.0	
3	15	2956	29.2	6.0	14.5	25.0	0.5	4.0	0.1	9.5		0.8	3.0	2.0	4.5	

Figure 6.3: Calling the data

If we only want to see the first few rows, we use the *head* method. Which is 5 by default.

```
drug.head()
```

	age	n	alcohol-use	alcol freque
0	12	2798	3.9	
1	13	2757	8.5	
2	14	2792	18.1	
3	15	2956	29.2	
4	16	3058	40.1	

5 rows × 28 columns

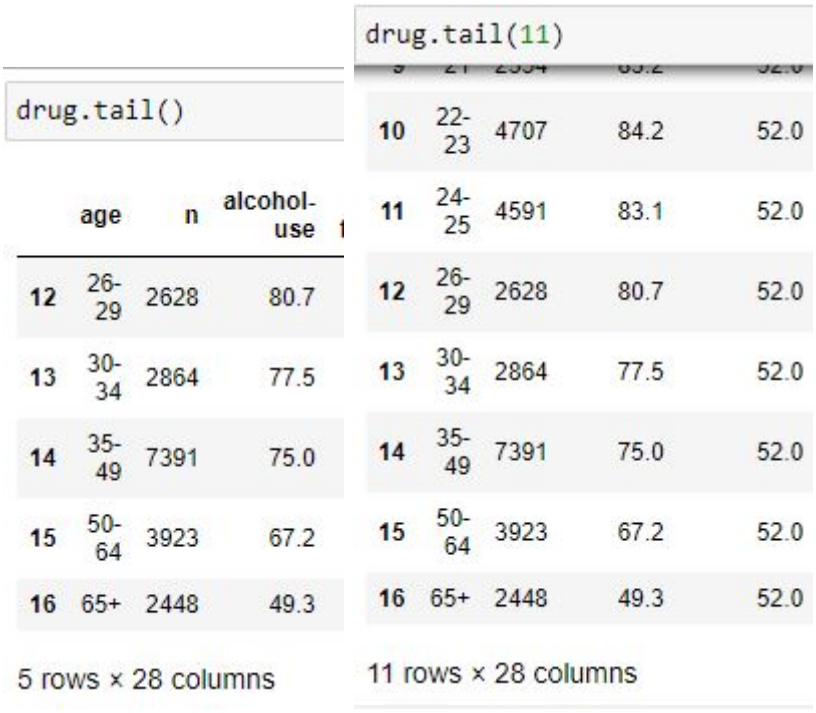
```
drug.head(10)
```

	age	n	alcohol-use	fre
0	12	2798	3.9	
1	13	2757	8.5	
2	14	2792	18.1	
3	15	2956	29.2	
4	16	3058	40.1	
5	17	3038	49.3	
6	18	2469	58.7	
7	19	2223	64.6	
8	20	2271	69.7	
9	21	2354	83.2	

10 rows × 28 columns

Figure 6.4: Using head method

Likewise, the *tail* method for last few rows.



```
drug.tail()
```

	age	n	alcohol-use
12	26-29	2628	80.7
13	30-34	2864	77.5
14	35-49	7391	75.0
15	50-64	3923	67.2
16	65+	2448	49.3

5 rows × 28 columns

```
drug.tail(11)
```

10	22-23	4707	84.2	52.0
11	24-25	4591	83.1	52.0
12	26-29	2628	80.7	52.0
13	30-34	2864	77.5	52.0
14	35-49	7391	75.0	52.0
15	50-64	3923	67.2	52.0
16	65+	2448	49.3	52.0

11 rows × 28 columns

Figure 6.5: Using tail method

The *index* method is used to examine index which could be handy to detect duplicates

```
In [11]: drug.index.has_duplicates
Out[11]: False
```

Figure 6.6: Using index method

We can use *shape* method to examine the dimensions of our data.

```
In [59]: drug.shape
Out[59]: (17, 28)
```

Figure 6.7: Using shape method

Bibliography

[1] M. Artin, *Algebra*. Pearson, 2010.