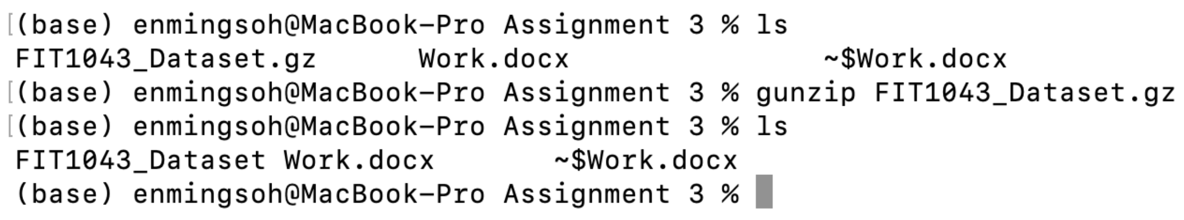
FIT 1043 Assignment 3

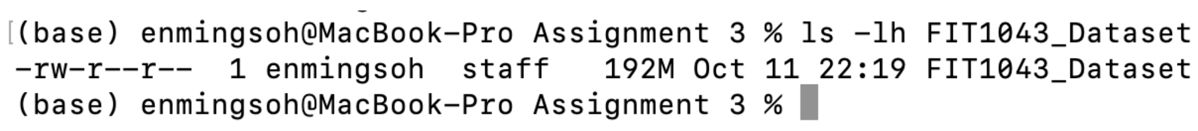
Unit Code: FIT1043
Name: Soh En Ming
Student ID: 32024975
Due Date: 14th October 2021

Part A: Investigating Sentiment140 Data using shell commands

Question 1

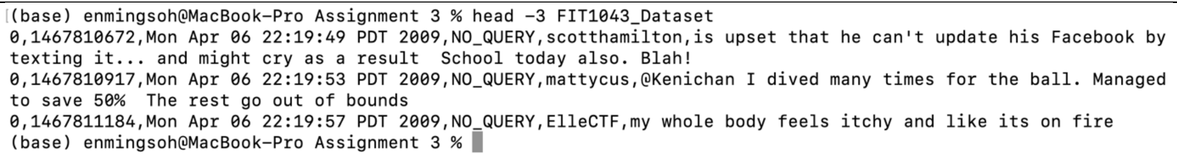
Explanation	To unzip/decompress the file, I will be using the command gunzip which basically stands for gnu unzip.
Code	`gunzip FIT1043_Dataset.gz`
Output	

```
[(base) enmingsoh@MacBook-Pro Assignment 3 % ls
 FIT1043_Dataset.gz      Work.docx                ~$Work.docx
[(base) enmingsoh@MacBook-Pro Assignment 3 % gunzip FIT1043_Dataset.gz
[(base) enmingsoh@MacBook-Pro Assignment 3 % ls
 FIT1043_Dataset Work.docx       ~$Work.docx
 (base) enmingsoh@MacBook-Pro Assignment 3 %
```

Explanation	To find what is the file size in megabytes, I used the command ls -lh which basically is list files and -lh is to find the size of the file in megabytes
Code	`ls -lh FIT1043_Dataset`
Output	
Output Explanation	The size of the file FIT1043_Dataset is 192 megabytes

```
[(base) enmingsoh@MacBook-Pro Assignment 3 % ls -lh FIT1043_Dataset
 -rw-r--r--  1 enmingsoh  staff   192M Oct 11 22:19 FIT1043_Dataset
 (base) enmingsoh@MacBook-Pro Assignment 3 %
```

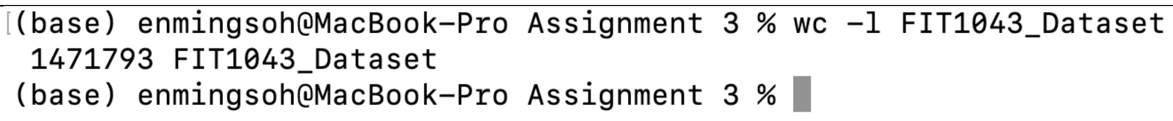
Question 2

Explanation	To find what is the delimiter used to separate the columns, I used the command head with -3 which shows the first 3 lines of the file.
Code	`head -3 FIT1043_Dataset`
Output	
Output Explanation	From the output above we can conlude that the delimeter that is used to separate the columns is a comma ",". For example lets use the 3rd line: 0,1467811184,Mon Apr 06 22:19:57 PDT 2009,NO_QUERY,ElleCTF,my whole body feels itchy and like its on fire. We can see that what seperates the columns are a comma ",".

```
[(base) enmingsoh@MacBook-Pro Assignment 3 % head -3 FIT1043_Dataset
0,1467810672,Mon Apr 06 22:19:49 PDT 2009,NO_QUERY,scotthamilton,is upset that he can't update his Facebook by
texting it... and might cry as a result  School today also. Blah!
0,1467810917,Mon Apr 06 22:19:53 PDT 2009,NO_QUERY,mattycus,@Kenichan I dived many times for the ball. Managed
to save 50%  The rest go out of bounds
0,1467811184,Mon Apr 06 22:19:57 PDT 2009,NO_QUERY,ElleCTF,my whole body feels itchy and like its on fire
(base) enmingsoh@MacBook-Pro Assignment 3 %
```

Question 3

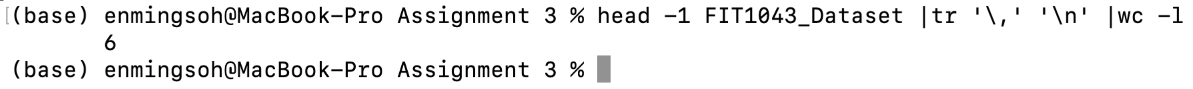
Explanation	In order to find how many rows there are in the file, we have to use the function wc which is normally used to find the number of lines, characters, words and bytes however in this case I will be using -l which finds specifically the number of lines/rows there are in the file.
Code	`wc -l FIT1043_Dataset`

Output	 `[(base) enmingsoh@MacBook-Pro Assignment 3 % wc -l FIT1043_Dataset`
	`  1471793 FIT1043_Dataset`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Explanation	From the output above, we can conclude that the number of rows there are in the file is 1471793

Question 4

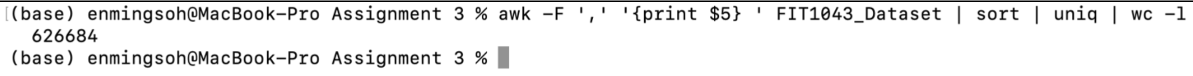
Explanation	To find the number of columns in the file, we have to use the command head -1 first to show the first line of the file then using a pipe for tr which is a command that replaces commas with a new line character then from there a last pipe which is wc -l which counts the number of lines there are. Each line being a column
Code	`head -1 FIT1043_Dataset |tr '\,' '\n' |wc -l`

Output	 `[(base) enmingsoh@MacBook-Pro Assignment 3 % head -1 FIT1043_Dataset |tr '\,' '\n' |wc -l`
	`       6`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Explanation	From the output above, we can conclude that the number of columns there are is 6.

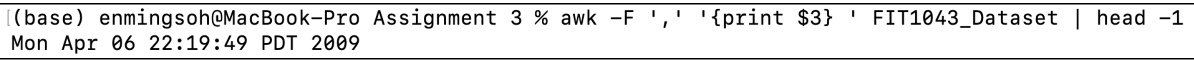
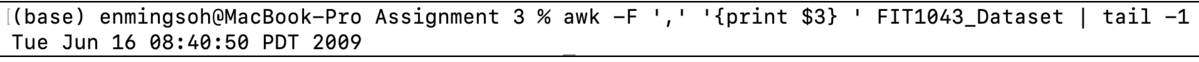
Question 5

Explanation	In order to find the number of unique users from the file, what I did was firstly use the command awk which is used to processing a text file one line at a time, then specifying the delimiter which in this case is "," and then {print $5} which means the 5th column then the file name, sort command to sort the lines according to order and removing duplicates at the same time then uniq command which is to give unique values and lastly wc -l which is used to count the total number of unique users. The code is shown below.
Code	`awk -F ',' '{print $5} ' FIT1043_Dataset | sort | uniq | wc -l`

Output	 `[(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $5} ' FIT1043_Dataset | sort | uniq | wc -l`
	`  626684`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

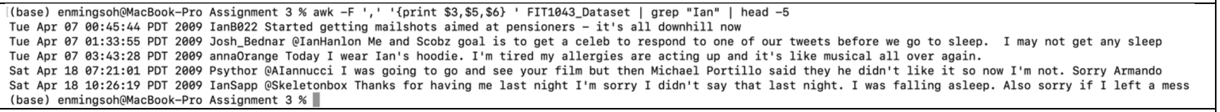
Output Explanation	From the output above, we can conclude that there is a total of 626684 unique users from the file.

Question 6

Explanation	Let’s assume the data is ordered by date in chronological order where the first date should be the first post at the first line and the last date should be the last post. From that understanding, we can just get the first date and last date to show the date range. In the code below, I’ve split it into 2 separate codes. One for the first post date and another one for the last post-date. What I’ve done for the code is to use awk to process one line at a time then print $3 to show which column (in this case it’s the date column), file name and a pipe for the command head -1 for the first line. Then for the last date I just switched head for tail. The code is shown below
Code 1	`awk -F ',' '{print $3} ' FIT1043_Dataset | head -1`
Output 1	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $3} ' FIT1043_Dataset | head -1` `Mon Apr 06 22:19:49 PDT 2009`
Code 2	`awk -F ',' '{print $3} ' FIT1043_Dataset | tail -1`
Output 2	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $3} ' FIT1043_Dataset | tail -1` `Tue Jun 16 08:40:50 PDT 2009`
Output Explanation	From the output above, we can give the date range for Twitter posts to be from Mon Apr 06 2009 to Tue Jun 16 2009. Additonal note: Although the answer above fits what the question requires, upon further analysing of the file, I have discovered that around the very middle of the file there is a further last post date of “Thu Jun 25 2009” which does not follow the format of chronologically ordered as if it was chronologically ordered, that date would be at the very bottom of the file. Instead it is in the middle suggesting it is ordered by sentiment then date. The revised date not according to the question specification but instead sentiement then date, the date range for Twitter posts would be from Mon Apr 06 2009 to Thu Jun 25 2009.

Question 7

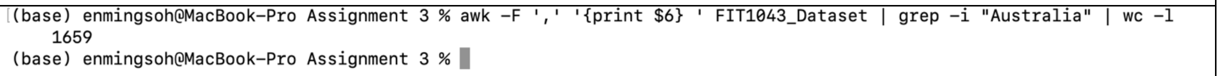
Explanation	To answer this question, what I did was implement awk command which checks line by line then print out columns 3, 5 and 6 since the question wants the date and time, the full message and user who mentioned Ian. Then I implemented a pipe then grep command which is used to filter out lines that include the word “Ian” then lastly a pipe then head command to show the first 5 lines. The code is shown below
Code	`awk -F ',' '{print $3,$5,$6} ' FIT1043_Dataset | grep "Ian" | head -5`

Output	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $3,$5,$6} ' FIT1043_Dataset | grep "Ian" | head -5`
	`Tue Apr 07 00:45:44 PDT 2009 IanB022 Started getting mailshots aimed at pensioners - it's all downhill now`
	`Tue Apr 07 01:33:55 PDT 2009 Josh_Bednar @IanHanlon Me and Scobz goal is to get a celeb to respond to one of our tweets before we go to sleep.  I may not get any sleep`
	`Tue Apr 07 03:43:28 PDT 2009 annaOrange Today I wear Ian's hoodie. I'm tired my allergies are acting up and it's like musical all over again.`
	`Sat Apr 18 07:21:01 PDT 2009 Psythor @AIannucci I was going to go and see your film but then Michael Portillo said they he didn't like it so now I'm not. Sorry Armando`
	`Sat Apr 18 10:26:19 PDT 2009 IanSapp @Skeletonbox Thanks for having me last night I'm sorry I didn't say that last night. I was falling asleep. Also sorry if I left a mess`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Explanation	From the output above, we can analyse the data shown and figure out that line 1 and 2 aren’t the answer. This is because the question specifally asked for “the first mention of the person with the name Ian and is not a retweet or reply to” however, although the first line the user has an Ian inside it doesn’t suit the criteria of the question as it want’s the first mention of the person Ian and also line 2 doesn’t suite the critera as it’s a reply to. This leaves the last 3 lines and if we look at line 3 we can see that it isn’t a retweet or a reply to and also doesn’t include Ian in the user but instead includes Ian in the text of the tweet which fits all the critera of the question. The answer for this question is Date and Time: Tue Apr 07 2009 03:43:28 The full message: Today I wear Ian's hoodie. I'm tired my allergies are acting up and it's like musical all over again. User that mentioned it: annaOrange This answer can be found in line 3 of the output.

Question 8

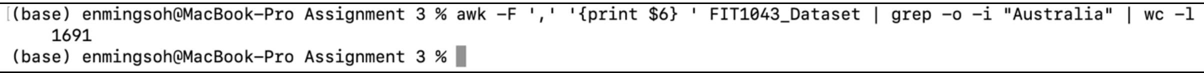
Explanation	To answer to find how many tweets have the word Australia, what I did was implement awk which reads every line, then specified the column by using print $6 then the filename then I used a pipe to create a new command called grep which is used to find the specific word. To add onto the command grep, I added -i which ignores the case like upper or lower case then added the word to find then lastly a pipe then wc command which calculates the number of lines and eventually gives us the number of tweets that have the word Australia. The code is shown below.
Code	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep -i "Australia" | wc -l`

Output	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep -i "Australia" | wc -l`
	`1659`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Explanation	From the output above, there is a total of 1659 tweets that have the word Australia within them.

Question 9

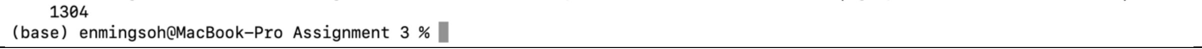
Explanation	Similar to question 8, I used awk to read every line then specified column 6 using print and then specified the filename. Then I used a pipe to use the command grep with -o which means only matching and -i which ignores the upper and lower case then specified the word. Lastly I created one last pipe for the command wc -l which calculates the number of lines. The code is shown below.
Code	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o -i "Australia" | wc -l`

Output	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o -i "Australia" | wc -l`
	`1691`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Explanation	From the output above, there is a total of 1691 times that have the word Australia within the column message.

Question 10

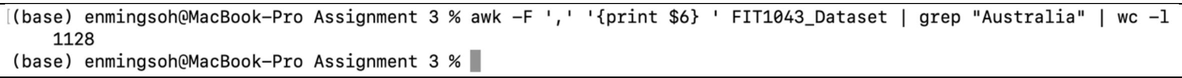
Explanation	For this question, we are not allowed to use any other word expect the exact word "Australia" with no case sensitivity. Therefore, to fulfil these parameters what I've done is firstly implemented the command awk which is useful for reading each line then specifying the column along with the filename, then using a pipe for another new command grep -o -i which is useful for finding the specified word with only matching word and no case sensitivity. Then specifying the word but with a slight difference compared to question 8 and 9. This time I used \b at the start of the word and the end which is used to make sure it is the exact word with no others like Australian or any words which have the characters Australia inside. Lastly, I used another pipe then the command wc -l which calculates the total number of lines calculated. The code is shown below.
Code	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o -i "\bAustralia\b" | wc -l`

Output	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o -i "\bAustralia\b" | wc -l`
	`1304`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

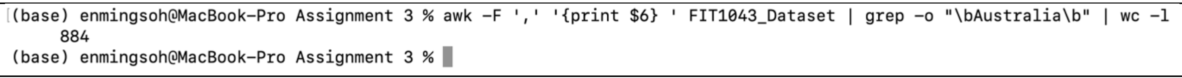
Output Explanation	From the output above, there is a total of 1304 which has the exact word "Australia" inside the message column.

Question 11

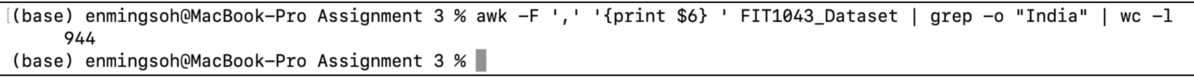
Explanation	Firstly, the definiton of what is meant for popular by me is whichever has the higher number of frequency or in this case the higher number of uses of the word used. Also in order to find which country is more popular, I will be doing 2 calculations for each country. The first calculation will be the number of tweets have the word India or Australia with case sensitivity (The reason I've chosen this is because I wanted to see how common people write the word India or Australia in tweets). The second calculation will be the number of times the exact word India/Australia has appeared in the message column with case sensitivity. Whichever country has the highest number compared to the other country for both calculations is the country that's more popular. In order to calculate this, I will be splitting this into 2 separate parts. One for Australia and one for India. Australia: For the first calculation code, to find how many tweets have the word Australia, what I did was implement awk which reads every line, then specified the column by using print $6 then the filename then I used a pipe to create a new command called grep which is used to find the specific word. Lastly a pipe then wc command which calculates the number of lines and eventually gives us the number of tweets that have the word Australia. The code is shown below at Code Australia 1. For the second calculation code, we are not allowed to use any other word expect the exact word "Australia" with case sensitivity. Therefore, to fulfil these parameters what I've done is firstly implemented the command awk which is useful for reading each line then specifying the column along with the filename, then using a pipe for another new command grep -o which is useful for finding the specified word with only matching word. Then specifying the word but with a slight difference compared to question 8 and 9. This time I used \b at the start of the word and the end which is used to make sure it is the exact word with no others like Australian or any words which have the characters Australia inside. Lastly, I used another pipe then the command wc -l which calculates the total number of lines calculated. The code is shown below at Code Australia 2. India: For the first calculation code, what I've done is used the code I used in code Australia 1 and replaced the "Australia" with "India". The code can be seen below at Code India 1. For the second calculation code, what I've done is used the code I used in code Australia 2 and replaced the "\bAustralia\b" with "\bIndia\b". The code can be seen below at Code India 2.
Code Australia 1	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep "Australia" | wc -l`

Output Australia 1	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep "Australia" | wc -l`
	`    1128`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

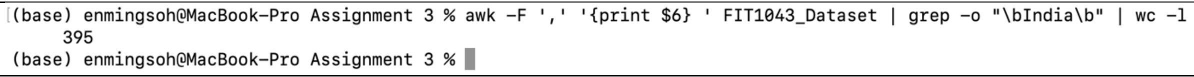
Output 1 Australia Explanation	From the output above, there is a total of 1128 tweets that have the word Australia within them.
Code Australia 2	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o "\bAustralia\b" | wc -l`

Output Australia 2	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o "\bAustralia\b" | wc -l`
	`     884`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output 2 Australia Explanation	From the output above, there is a total of 884 which has the exact word "Australia" inside the message column.
Code India 1	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o "India" | wc -l`

Output India 1	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o "India" | wc -l`
	`     944`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

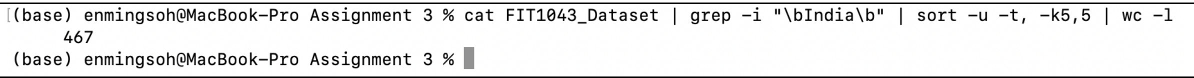
Output 1 India Explanation	From the output above, there is a total of 944 tweets that have the word India within them.
Code India 2	`awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o "\bIndia\b" | wc -l`

Output India 2	 `(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '{print $6} ' FIT1043_Dataset | grep -o "\bIndia\b" | wc -l`
	`     395`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output 2 India Explanation	From the output above, there is a total of 395 which has the exact word "India" inside the message column.
Final Explanation for which is more popular	As explanined above, in order to find which country is more popular, there has been 2 calculations made for each country. From there we will check which country is more popular for each calculation. For calculation 1, Australia has a total of 1128 tweets which is more than India's total tweets of 944. Therefore, for calculation 1 Australia is more popular than India. For calculation 2, Australia has a total of 884 which has the exact word of "Australia" which in this case is more than India's 395 exact word of "India". Therefore, for calculation 2 Australia is once again more popular. In conclusion, the country which is more popular is Australia since it has more tweets and exact words for Australia compared to India with case sensitivity.

Question 12

Explanation	For this question it asked to select the posts which has the country India in the twitter message and then also include how many unique users mentioned India. In order to solve this question, I started with cat to open up the dataset then I used a pipe for another command which is grep -i. The purpose for this command is to filter but since I used -i it is no longer case sensitive and I specified the word with \b which means it can only be that word. After the command grep I used a pipe for command sort -u -t, -k5,5 which is used to sort alphabetically with numbers first however, I modified the command with -u which is unique, -t which is for comma field separator and lastly -k5,5 which is to specify where the sort command should be for which in this case is 5. Then lastly, one more pipe for the command wc -l to count the number of lines.
Code	`cat FIT1043_Dataset | grep -i "\bIndia\b" | sort -u -t, -k5,5 | wc -l`

Output	 `(base) enmingsoh@MacBook-Pro Assignment 3 % cat FIT1043_Dataset | grep -i "\bIndia\b" | sort -u -t, -k5,5 | wc -l`
	`     467`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Explanation	From the output above, there is a total of 467 unique users with twitter messages that has mentioned the exact word India.

Question 13

For Australia

Explanation

The code below is for Australia. In order to solve this question which requires to find the total amount of positive, neutral and negative tweets for Australia and India, I have devised the code which is shown below.

The code will have 3 parts to it, the first part for total number of Negative tweets, the second part will contain the total number of neutral tweets and lastly the third part which will contain the total number of positive tweets

Firstly, I've started with printf command which is basically used to display a string which is "Negative, " and then > sentiment-australia.csv which means the newly created file where the string will be stored in. Then a pipe for the command awk since it's helpful in reading every line then I specified the delimiter, which is comma, then the columns which awk will read which is 1 and 6 then the specified filename. After that another pipe will be made for the command grep -i which will filter out according to the word specified which in this case is "Australia" and then using -i so that it ignores any case sensitivity. Then there's another pipe for command grep which in this case filters out values which have a "0" inside then another pipe for wc -l which counts the number of lines and lastly sed command which helps remove any spaces before the value and >> command which is used to append the values into the csv file sentiment-australia.

For the second part, I've just reused the first code but changed a few words which are Negative to Neutral and grep "0" to grep "2" since we're finding neutral now.

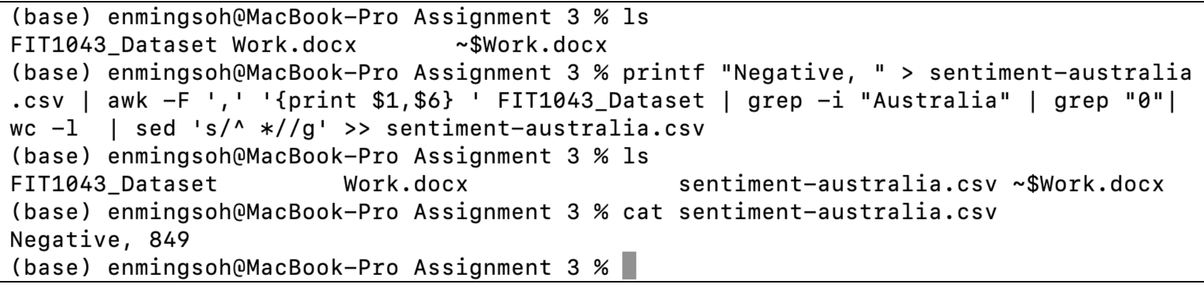
For the third part, I've just reused the first code but changed a few words which are Negative to Positive and grep "0" to grep "4" since we're finding positive now.

To check if the code has successfully created a new file as well as appended the code into the new csv file, I will be using ls firstly to show the file has been created and cat to show the contents of the file.

Code Pt 1

```
printf "Negative, " > sentiment-australia.csv | awk -F ',' '{print
$1,$6} ' FIT1043_Dataset | grep -i "Australia" | grep "0"| wc -l |
sed 's/^ *//g' >> sentiment-australia.csv
```

Output Pt 1



```
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset Work.docx       ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % printf "Negative, " > sentiment-australia
.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "Australia" | grep "0"|
wc -l  | sed 's/^ *//g' >> sentiment-australia.csv
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset         Work.docx               sentiment-australia.csv ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % cat sentiment-australia.csv
Negative, 849
(base) enmingsoh@MacBook-Pro Assignment 3 %
```

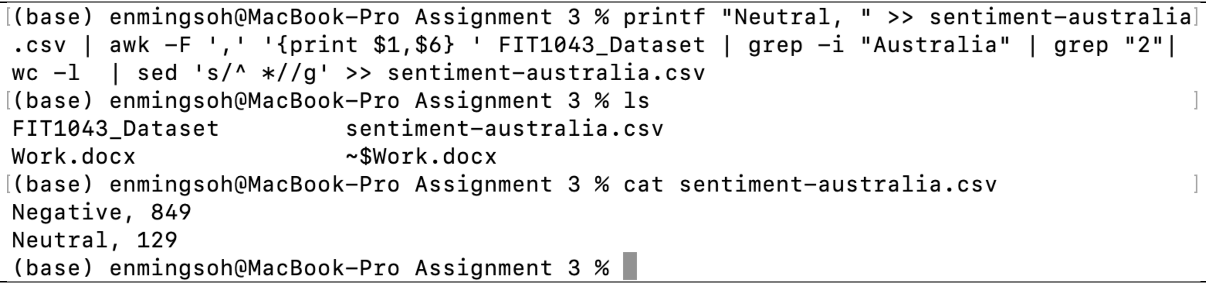
Output Pt 1 Explanation

From the output above, there is a total number of 849 users who had negative tweets for Australia.

Code Pt 2

```
printf "Neutral, " >> sentiment-australia.csv | awk -F ',' '{print
$1,$6} ' FIT1043_Dataset | grep -i "Australia" | grep "2"| wc -l |
sed 's/^ *//g' >> sentiment-australia.csv
```

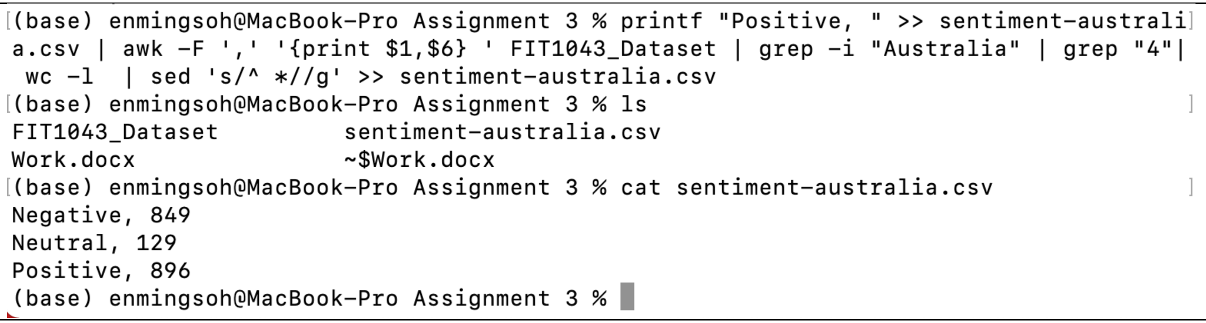
Output Pt 2



```
(base) enmingsoh@MacBook-Pro Assignment 3 % printf "Neutral, " >> sentiment-australia
.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "Australia" | grep "2"|
wc -l  | sed 's/^ *//g' >> sentiment-australia.csv
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset         sentiment-australia.csv
Work.docx               ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % cat sentiment-australia.csv
Negative, 849
Neutral, 129
(base) enmingsoh@MacBook-Pro Assignment 3 %
```

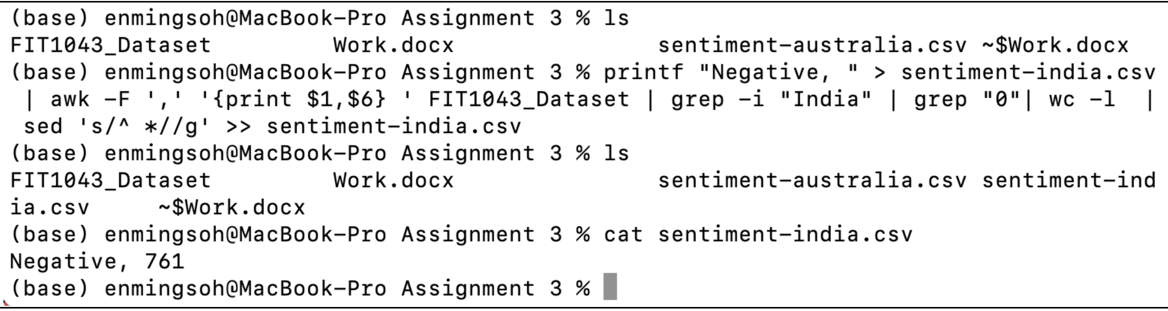
Output Pt 2 Explanation

From the output above, there is a total number of 129 users who had neutral tweets for Australia.

Code Pt 3	`printf "Positive, " >> sentiment-australia.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "Australia" | grep "4"| wc -l  | sed 's/^ *//g' >> sentiment-australia.csv`
Output Pt 3	
Output Pt 3 Explanation	From the output above, there is a total number of 896 users who had positive tweets for Australia.

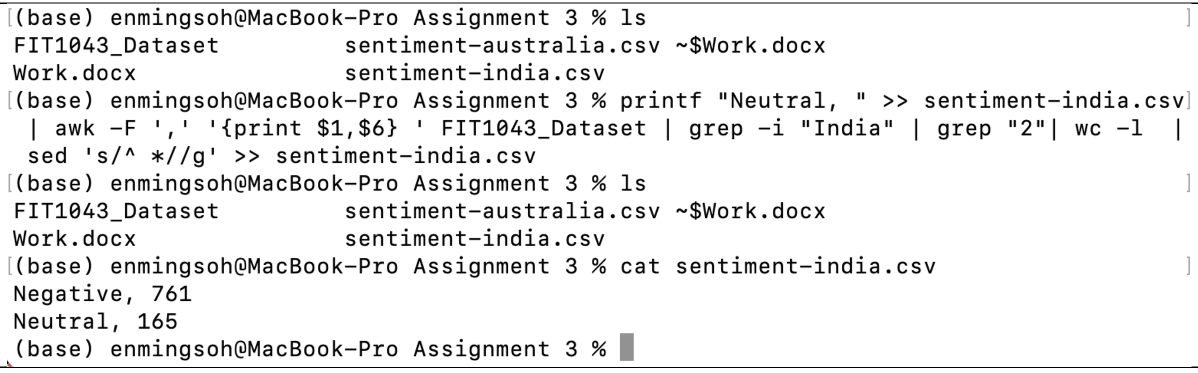
```
(base) enmingsoh@MacBook-Pro Assignment 3 % printf "Positive, " >> sentiment-australi
a.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "Australia" | grep "4"|
 wc -l  | sed 's/^ *//g' >> sentiment-australia.csv
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset         sentiment-australia.csv
Work.docx               ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % cat sentiment-australia.csv
Negative, 849
Neutral, 129
Positive, 896
(base) enmingsoh@MacBook-Pro Assignment 3 %
```

For India

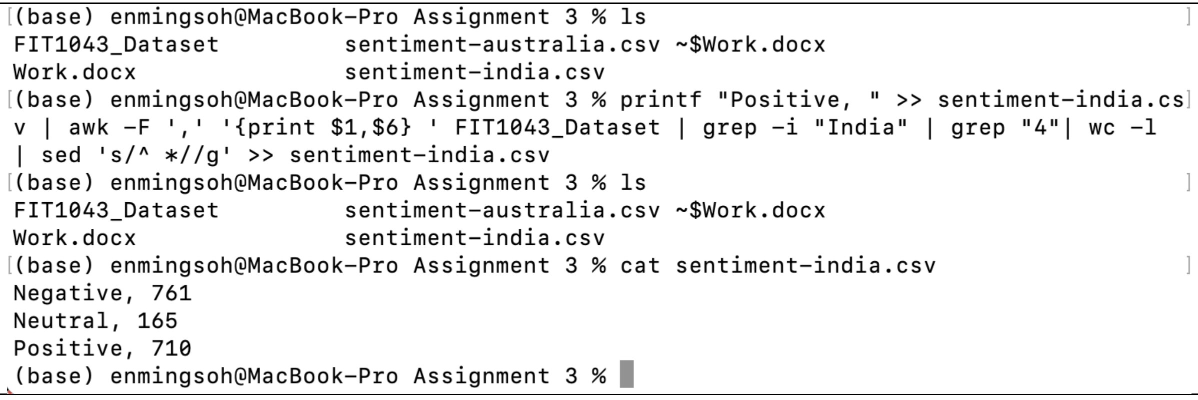
Explanation	The code below is for India. In order to solve this question which requires to find the total amount of positive, neutral and negative tweets for Australia and India, I have devised the code which is shown below. The code will have 3 parts to it, the first part for total number of Negative tweets, the second part will contain the total number of neutral tweets and lastly the third part which will contain the total number of positive tweets Firstly, I've started with printf command which is basically used to display a string which is "Negative, " and then > sentiment-india.csv which means the newly created file where the string will be stored in. Then a pipe for the command awk since it's helpful in reading every line then I specified the delimiter, which is comma, then the columns which awk will read which is 1 and 6 then the specified filename. After that another pipe will be made for the command grep -i which will filter out according to the word specified which in this case is "India" and then using -i so that it ignores any case sensitivity. Then there's another pipe for command grep which in this case filters out values which have a "0" inside then another pipe for wc -l which counts the number of lines and lastly sed command which helps remove any spaces before the value and >> command which is used to append the values into the csv file sentiment-india. For the second part, I've just reused the first code but changed a few words which are Negative to Neutral and grep "0" to grep "2" since we're finding neutral now. For the third part, I've just reused the first code but changed a few words which are Negative to Positive and grep "0" to grep "4" since we're finding positive now. To check if the code has successfully created a new file as well as appended the code into the new csv file, I will be using ls firstly to show the file has been created and cat to show the contents of the file.
Code Pt 1	`printf "Negative, " > sentiment-india.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "India" | grep "0"| wc -l  | sed 's/^ *//g' >> sentiment-india.csv`
Output Pt 1	
Output Pt 1 Explanation	From the output above, there is a total number of 761 users who had negative tweets for India.

```
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset         Work.docx               sentiment-australia.csv ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % printf "Negative, " > sentiment-india.csv
 | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "India" | grep "0"| wc -l  |
 sed 's/^ *//g' >> sentiment-india.csv
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset         Work.docx               sentiment-australia.csv sentiment-ind
ia.csv     ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % cat sentiment-india.csv
Negative, 761
(base) enmingsoh@MacBook-Pro Assignment 3 %
```

Code Pt 2	`printf "Neutral, " >> sentiment-india.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "India" | grep "2"| wc -l  | sed 's/^ *//g' >> sentiment-india.csv`

Output Pt 2	 `(base) enmingsoh@MacBook-Pro Assignment 3 % ls`
	`FIT1043_Dataset         sentiment-australia.csv ~$Work.docx`
	`Work.docx               sentiment-india.csv`
	`(base) enmingsoh@MacBook-Pro Assignment 3 % printf "Neutral, " >> sentiment-india.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "India" | grep "2"| wc -l  | sed 's/^ *//g' >> sentiment-india.csv`
	`(base) enmingsoh@MacBook-Pro Assignment 3 % ls`
	`FIT1043_Dataset         sentiment-australia.csv ~$Work.docx`
	`Work.docx               sentiment-india.csv`
	`(base) enmingsoh@MacBook-Pro Assignment 3 % cat sentiment-india.csv`
	`Negative, 761`
	`Neutral, 165`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

Output Pt 2 Explanation	From the output above, there is a total number of 165 users who had neutral tweets for India.
Code Pt 3	`printf "Positive, " >> sentiment-india.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "India" | grep "4"| wc -l  | sed 's/^ *//g' >> sentiment-india.csv`

Output Pt 3	 `(base) enmingsoh@MacBook-Pro Assignment 3 % ls`
	`FIT1043_Dataset         sentiment-australia.csv ~$Work.docx`
	`Work.docx               sentiment-india.csv`
	`(base) enmingsoh@MacBook-Pro Assignment 3 % printf "Positive, " >> sentiment-india.csv | awk -F ',' '{print $1,$6} ' FIT1043_Dataset | grep -i "India" | grep "4"| wc -l | sed 's/^ *//g' >> sentiment-india.csv`
	`(base) enmingsoh@MacBook-Pro Assignment 3 % ls`
	`FIT1043_Dataset         sentiment-australia.csv ~$Work.docx`
	`Work.docx               sentiment-india.csv`
	`(base) enmingsoh@MacBook-Pro Assignment 3 % cat sentiment-india.csv`
	`Negative, 761`
	`Neutral, 165`
	`Positive, 710`
	`(base) enmingsoh@MacBook-Pro Assignment 3 %`

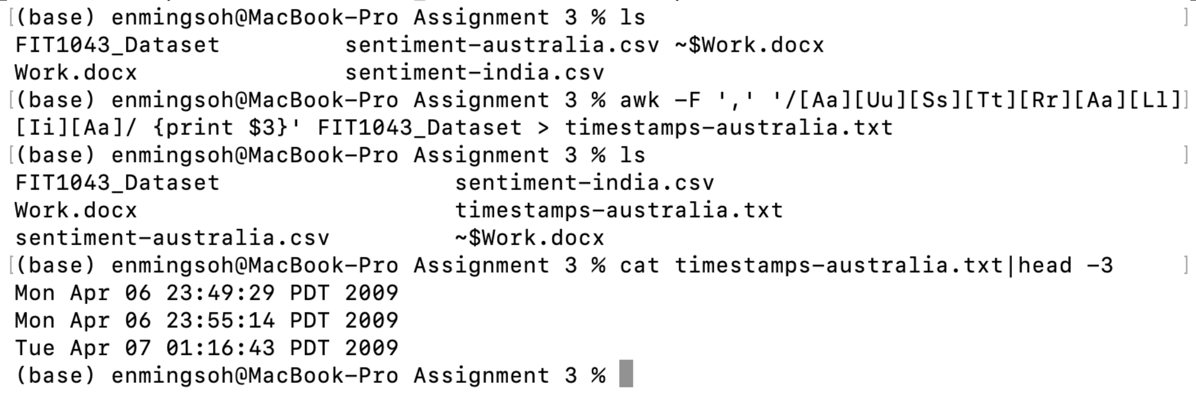
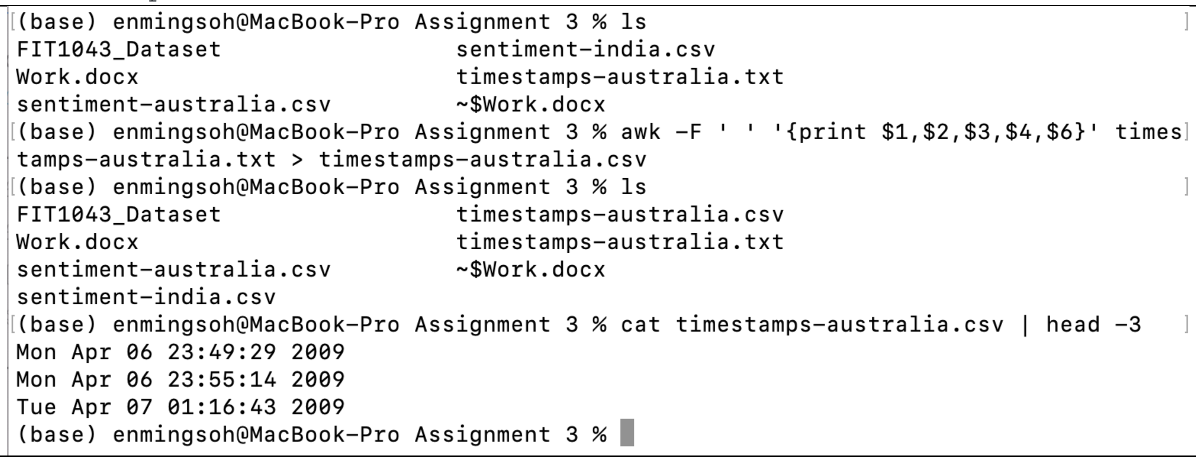
Output Pt 3 Explanation	From the output above, there is a total number of 710 users who had positive tweets for India.

Conclusion Explanation for both countries Australia and India	The country that has more postivie tweets related to it is Australia with 896 positive tweets from users. Compared to India which only has 710 positive tweets from users. In terms of justification, I think my answer is justified. This can in 2 ways. Firstly, the question asked which country seems to have more positive tweets related to them and from the data we can clearly see that Australia has 896 positive tweets which is more than India's 710 positive tweets. The second way is using ratio of positive to negative. Let's say india has more postive tweets then negative tweets and the ratio between them is much higher than Australia then we could say that although Australia has more positive tweets the ratio for postive to negative is much lesser than India however, that isn't the case as Australia has more positive tweets then negative tweets whilst India has less positive tweets compared to negative. Therefore, we can conclude that Australia is the country that seems to have more positive tweets.

Part B: R, Graphing and Machine Learning

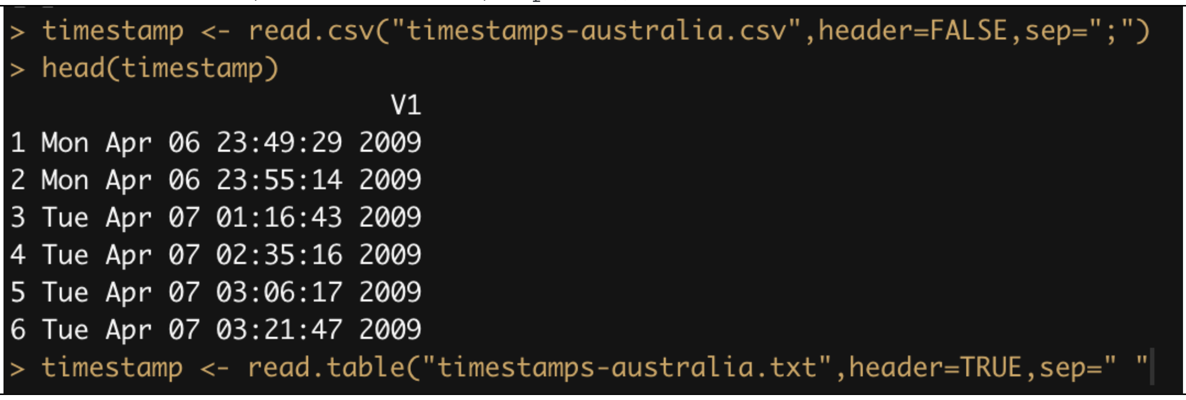
Question 1

i)

Explanation	To answer this question, what I've done is used the awk command to find the delimiter of the file, then the word which is Australia. However, to use awk to find the word Australia without any case sensitivity, I had to use both the upper case and lower case of that letter. After that I specified column 3 then the filename to collect the data where it will be stored in timestamps-australia.txt.
Code 1	`awk -F ',' '/[Aa][Uu][Ss][Tt][Rr][Aa][Ll][Ii][Aa]/ {print $3}' FIT1043_Dataset > timestamps-australia.txt`
Output1	 [(base) enmingsoh@MacBook-Pro Assignment 3 % ls] FIT1043_Dataset sentiment-australia.csv ~$Work.docx Work.docx sentiment-india.csv [(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ',' '/[Aa][Uu][Ss][Tt][Rr][Aa][Ll]] [Ii][Aa]/ {print $3}' FIT1043_Dataset > timestamps-australia.txt [(base) enmingsoh@MacBook-Pro Assignment 3 % ls] FIT1043_Dataset sentiment-india.csv Work.docx timestamps-australia.txt sentiment-australia.csv ~$Work.docx [(base) enmingsoh@MacBook-Pro Assignment 3 % cat timestamps-australia.txt|head -3] Mon Apr 06 23:49:29 PDT 2009 Mon Apr 06 23:55:14 PDT 2009 Tue Apr 07 01:16:43 PDT 2009 (base) enmingsoh@MacBook-Pro Assignment 3 %
Output 1 Explanation	The output is a file called timestamps-australia.txt which contains the column 3 which is the date of the tweet and also the date of the tweets are all from users who has Australia mentioned in their messages.
Explanation	From the file generated in the first code we can see that there's a column that contains the zone of the sender however, in this case it isn't really necessary as later in ii) it will require to use strptime() function which the zone of the sender won't affect it much therefore, I've decided to create a code to remove the column so it's left only with column 1, 2, 3, 4, 6. The code I used is awk function to print only columns 1, 2, 3, 4, 6 from the file timestamps-australia.txt then rewriting it with the removed column in a csv file.
Code 2	`awk -F ' ' '{print $1,$2,$3,$4,$6}' timestamps-australia.txt > timestamps-australia.csv`
Output 2	 [(base) enmingsoh@MacBook-Pro Assignment 3 % ls] FIT1043_Dataset sentiment-india.csv Work.docx timestamps-australia.txt sentiment-australia.csv ~$Work.docx [(base) enmingsoh@MacBook-Pro Assignment 3 % awk -F ' ' '{print $1,$2,$3,$4,$6}' times] tamps-australia.txt > timestamps-australia.csv [(base) enmingsoh@MacBook-Pro Assignment 3 % ls] FIT1043_Dataset timestamps-australia.csv Work.docx timestamps-australia.txt sentiment-australia.csv ~$Work.docx sentiment-india.csv [(base) enmingsoh@MacBook-Pro Assignment 3 % cat timestamps-australia.csv | head -3] Mon Apr 06 23:49:29 2009 Mon Apr 06 23:55:14 2009 Tue Apr 07 01:16:43 2009 (base) enmingsoh@MacBook-Pro Assignment 3 %
Output 2 Explanation	From the output above, there is no longer a column which contains the time zone.

ii)

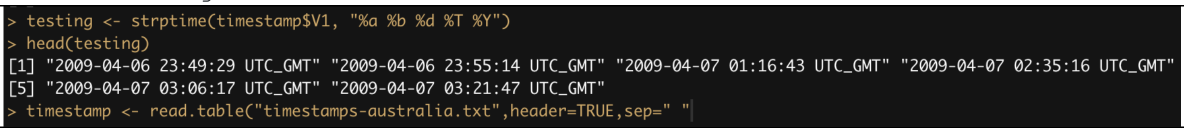
Explanation	For this part of the question, in order to read the file created earlier in bash, I've decided to code it by firstly read.csv function which works by firstly listing the csv filename then a comma for whether a header is present or not which in this case I choose FALSE then a sep which stands for field separator which was chosen to be ";" since ";" is commonly used for csv. The function read.csv() will be stored in timestamp as seen in the beginning of the code. Head(timestamp) is used to show that the csv file has been stored in timestamp.
Code 1	`timestamp <- read.csv("timestamps-australia.csv",header=FALSE,sep=";")`

Output 1	
	`> timestamp <- read.csv("timestamps-australia.csv",header=FALSE,sep=";")`
	`> head(timestamp)`
	`                         V1`
	`1 Mon Apr 06 23:49:29 2009`
	`2 Mon Apr 06 23:55:14 2009`
	`3 Tue Apr 07 01:16:43 2009`
	`4 Tue Apr 07 02:35:16 2009`
	`5 Tue Apr 07 03:06:17 2009`
	`6 Tue Apr 07 03:21:47 2009`
	`> timestamp <- read.table("timestamps-australia.txt",header=TRUE,sep=" "`

Output 1 Explanation	As seen in the output above, the file timestamps-australia.csv has been successfully stored in timestamp.

Explanation	This is a continuation from the code earlier. This code will be using the function strptime() which basically is used to convert text values from files to dates and calendar times. How I implemented this code is through using the function strptime() then inside the function I stated the variable which contains the text values which in this case is timestamp from earlier than $V1 which is used so that the values won't come out as NA. After that I added "%a %b %d %T %Y" which is basically each column of the data in timestamp but each %alphabet used has a special purpose for that specific column. First is %a which is used for texts which has an abbreviated weekday name, then %b which is for texts which has abbreviated month name, then %d which is for day of the month, then %T which is for the time and lastly %Y for the year. The function strptime is stored in testing. To check whether the function has worked, I've used the function head to check the first couple lines.

Code 2	`testing <- strptime(timestamp$V1, "%a %b %d %T %Y")`
	`head(testing)`

Output 2	
	`> testing <- strptime(timestamp$V1, "%a %b %d %T %Y")`
	`> head(testing)`
	`[1] "2009-04-06 23:49:29 UTC_GMT" "2009-04-06 23:55:14 UTC_GMT" "2009-04-07 01:16:43 UTC_GMT" "2009-04-07 02:35:16 UTC_GMT"`
	`[5] "2009-04-07 03:06:17 UTC_GMT" "2009-04-07 03:21:47 UTC_GMT"`
	`> timestamp <- read.table("timestamps-australia.txt",header=TRUE,sep=" "`

Output 2 Explanation	As seen in the output above, testing has stored the new values created by the strptime function.

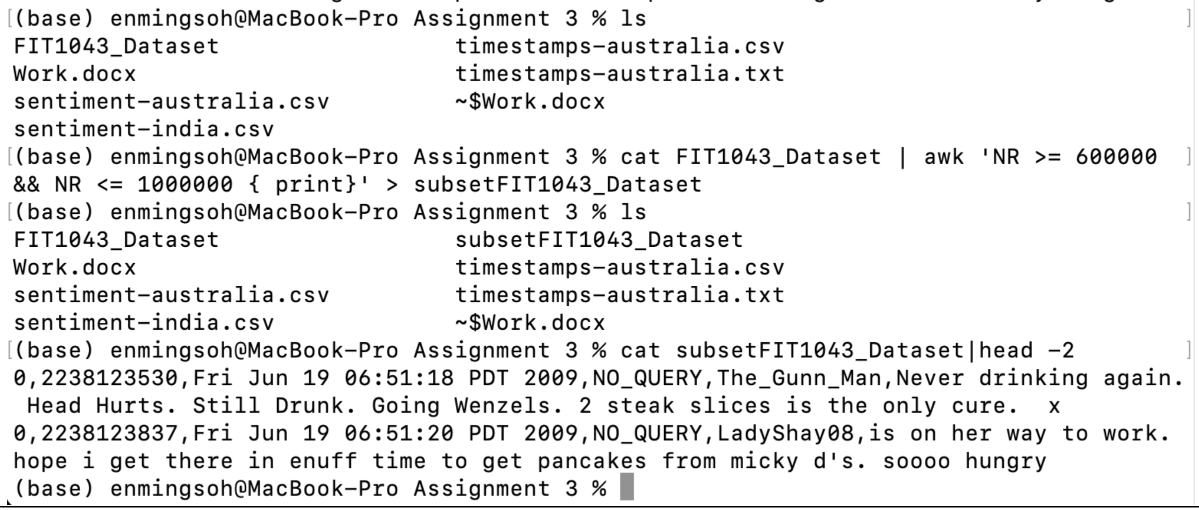
iii)

Explanation	This is a line graph which shows how many tweets have been posted during that day. What I've decided to code is t as the dates, z as the number of days of the week and then a plot function which will be used to plot the line graph with specific details like the line colour orange or the x label name, y label name, name of the graph.

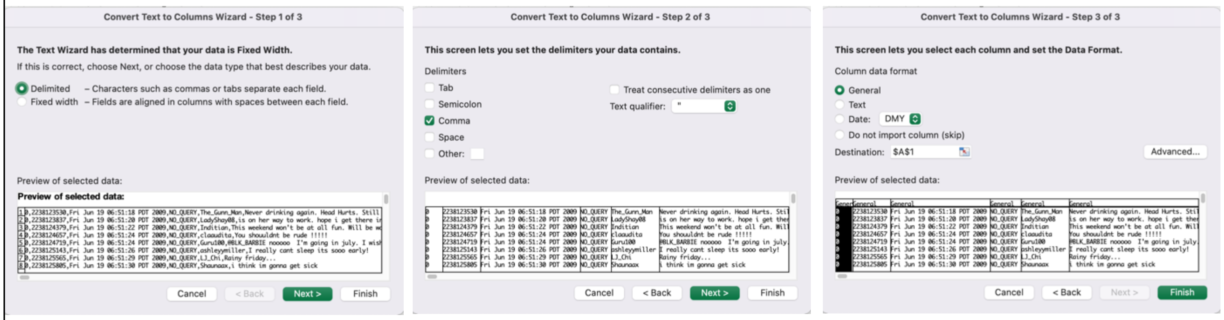
Code	`t = testing`
	`z = 0:7`
	`plot(t,z, type="l", col="orange", lwd=5, xlab="Dates",`
	`     ylab="Number of tweets posted during that day",`
	`     main="Line graph showing the number of tweets that was tweeted during that day.")`

Output Explanation	A line graph has been created where the x axis shows each day and the y axis shows the number of tweets that were posted during that day

Question 2

Explanation	In question 2 it stated that I can conduct any pre-processing, therefore I've decided to create a subset of the data so that I won't be encountering so much memory problems. In the code below, I've used cat then the file FIT1043_Dataset so that I can temporary open the file then using a pipe awk, I can use the characters NR to filter from 600000 row to 1000000 rows along with print so that it can print all the columns into the new file subsetFIT1043_Dataset using >.
Code	`cat FIT1043_Dataset | awk 'NR >= 600000 && NR <= 1000000 { print}' > subsetFIT1043_Dataset`
Output	
Output Explanation	As seen in the output above, the new subset of FIT1043_Dataset has been created.

```
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset                 timestamps-australia.csv
Work.docx                       timestamps-australia.txt
sentiment-australia.csv         ~$Work.docx
sentiment-india.csv
(base) enmingsoh@MacBook-Pro Assignment 3 % cat FIT1043_Dataset | awk 'NR >= 600000
&& NR <= 1000000 { print}' > subsetFIT1043_Dataset
(base) enmingsoh@MacBook-Pro Assignment 3 % ls
FIT1043_Dataset                 subsetFIT1043_Dataset
Work.docx                       timestamps-australia.csv
sentiment-australia.csv         timestamps-australia.txt
sentiment-india.csv             ~$Work.docx
(base) enmingsoh@MacBook-Pro Assignment 3 % cat subsetFIT1043_Dataset|head -2
0,2238123530,Fri Jun 19 06:51:18 PDT 2009,NO_QUERY,The_Gunn_Man,Never drinking again.
 Head Hurts. Still Drunk. Going Wenzels. 2 steak slices is the only cure.  x
0,2238123837,Fri Jun 19 06:51:20 PDT 2009,NO_QUERY,LadyShay08,is on her way to work.
hope i get there in enuff time to get pancakes from micky d's. soooo hungry
(base) enmingsoh@MacBook-Pro Assignment 3 % 
```

Explanation	Another pre-processing that I've done is copy all the code from the subsetFIT1043_Dataset file into an excel spreadsheet where I pasted all the values then used the button text to columns that can be found in data. This is in order to remove the commas and make each section a column so that it can be easier to be used in r later.
Picture	

Convert Text to Columns Wizard - Step 1 of 3

The Text Wizard has determined that your data is Fixed Width.

If this is correct, choose Next, or choose the data type that best describes your data.

[✓ Delimited] – Characters such as commas or tabs separate each field.
Fixed width – Fields are aligned in columns with spaces between each field.

Preview of selected data:

```
1 0,2238123530,Fri Jun 19 06:51:18 PDT 2009,NO_QUERY,The_Gunn_Man,Never drinking again. Head Hurts. Still
2 0,2238123837,Fri Jun 19 06:51:20 PDT 2009,NO_QUERY,LadyShay08,is on her way to work. hope i get there in
3 0,2238124379,Fri Jun 19 06:51:22 PDT 2009,NO_QUERY,Indition,This weekend won't be at all fun. Will be w
4 0,2238124657,Fri Jun 19 06:51:24 PDT 2009,NO_QUERY,claaudita,You shouuldnt be rude !!!!!
5 0,2238124719,Fri Jun 19 06:51:24 PDT 2009,NO_QUERY,Guru100,@BLK_BARBIE nooooo  I'm going in july. I wish
6 0,2238125143,Fri Jun 19 06:51:26 PDT 2009,NO_QUERY,ashleyymiller,I really cant sleep its sooo early!
7 0,2238125565,Fri Jun 19 06:51:29 PDT 2009,NO_QUERY,LJ_Chi,Rainy friday...
8 0,2238125805,Fri Jun 19 06:51:30 PDT 2009,NO_QUERY,Shaunaax,i think im gonna get sick
```

Convert Text to Columns Wizard - Step 2 of 3

This screen lets you set the delimiters your data contains.

Delimiters: Tab, Semicolon, [✓ Comma], Space, Other:
Treat consecutive delimiters as one
Text qualifier: "

Preview of selected data:

```
0  2238123530 Fri Jun 19 06:51:18 PDT 2009 NO_QUERY The_Gunn_Man  Never drinking again. Head Hurts. Stil
0  2238123837 Fri Jun 19 06:51:20 PDT 2009 NO_QUERY LadyShay08    is on her way to work. hope i get ther
0  2238124379 Fri Jun 19 06:51:22 PDT 2009 NO_QUERY Indition      This weekend won't be at all fun. Will
0  2238124657 Fri Jun 19 06:51:24 PDT 2009 NO_QUERY claaudita     You shouuldnt be rude !!!!!
0  2238124719 Fri Jun 19 06:51:24 PDT 2009 NO_QUERY Guru100       @BLK_BARBIE nooooo  I'm going in july.
0  2238125143 Fri Jun 19 06:51:26 PDT 2009 NO_QUERY ashleyymiller I really cant sleep its sooo early!
0  2238125565 Fri Jun 19 06:51:29 PDT 2009 NO_QUERY LJ_Chi        Rainy friday...
0  2238125805 Fri Jun 19 06:51:30 PDT 2009 NO_QUERY Shaunaax      i think im gonna get sick
```

Convert Text to Columns Wizard - Step 3 of 3

This screen lets you select each column and set the Data Format.

Column data format: [✓ General], Text, Date: DMY, Do not import column (skip)
Destination: A1

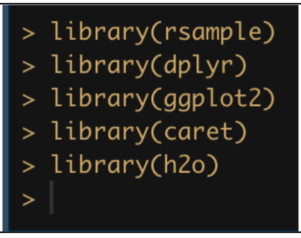
Preview of selected data:

```
Gener General    General                      General  General       General
0     2238123530 Fri Jun 19 06:51:18 PDT 2009 NO_QUERY The_Gunn_Man  Never drinking again. Head Hurts. Stil
0     2238123837 Fri Jun 19 06:51:20 PDT 2009 NO_QUERY LadyShay08    is on her way to work. hope i get ther
0     2238124379 Fri Jun 19 06:51:22 PDT 2009 NO_QUERY Indition      This weekend won't be at all fun. Will
0     2238124657 Fri Jun 19 06:51:24 PDT 2009 NO_QUERY claaudita     You shouuldnt be rude !!!!!
0     2238124719 Fri Jun 19 06:51:24 PDT 2009 NO_QUERY Guru100       @BLK_BARBIE nooooo  I'm going in july.
0     2238125143 Fri Jun 19 06:51:26 PDT 2009 NO_QUERY ashleyymiller I really cant sleep its sooo early!
0     2238125565 Fri Jun 19 06:51:29 PDT 2009 NO_QUERY LJ_Chi        Rainy friday...
0     2238125805 Fri Jun 19 06:51:30 PDT 2009 NO_QUERY Shaunaax      i think im gonna get sick
```

Output Explanation	After the conversion from text to columns has been done, it will be seperated into columns and then I will save the csv as subsetFIT1043_Dataset.csv

Explanation 1	After creating the new csv, it's time to move into R Studio where I will determine the datasets for training and testing, then implement naïve bayes to train and build the model, conduct model testing, and lastly interpret the output confusion matrix. Since the code is quite long, I will be splitting it up into several parts so I can explain it much better. If you want to look at the full code, it will be at the bottom after all the several codes. The websites that I've referred to and referenced from are https://uc-r.github.io/naive_bayes https://www.edureka.co/blog/naive-bayes-in-r/ The links above, I've used as a base for my code and also have used some of their codes in order to get my naïve bayes algorithm in R programming to work. Firstly, let's begin with importing libraries. In R programming, there's options to import libraries like how we can in Python which are very helpful for certain purposes. To import libraries, you have to firstly install them. To install libraries you have to use the syntax `install.packages("rsample")` which it will then start to install after you run the code. This syntax install.packages works for every single library all you need to do is to put the library name within the "". After installing the libraries, you should be run the libraries in the code below. The rsample library is helpful for data splitting, the dplyr library is helpful for data transformation, the library ggplot2 is helpful for plotting/visualization, the library caret is helpful for model training and lastly the library h2o is useful for implementation. The code is shown below.

Code 1	`library(rsample)`
	`library(dplyr)`
	`library(ggplot2)`
	`library(caret)`
	`library(h2o)`

Output 1	
	`> library(rsample)`
	`> library(dplyr)`
	`> library(ggplot2)`
	`> library(caret)`
	`> library(h2o)`
	`>`

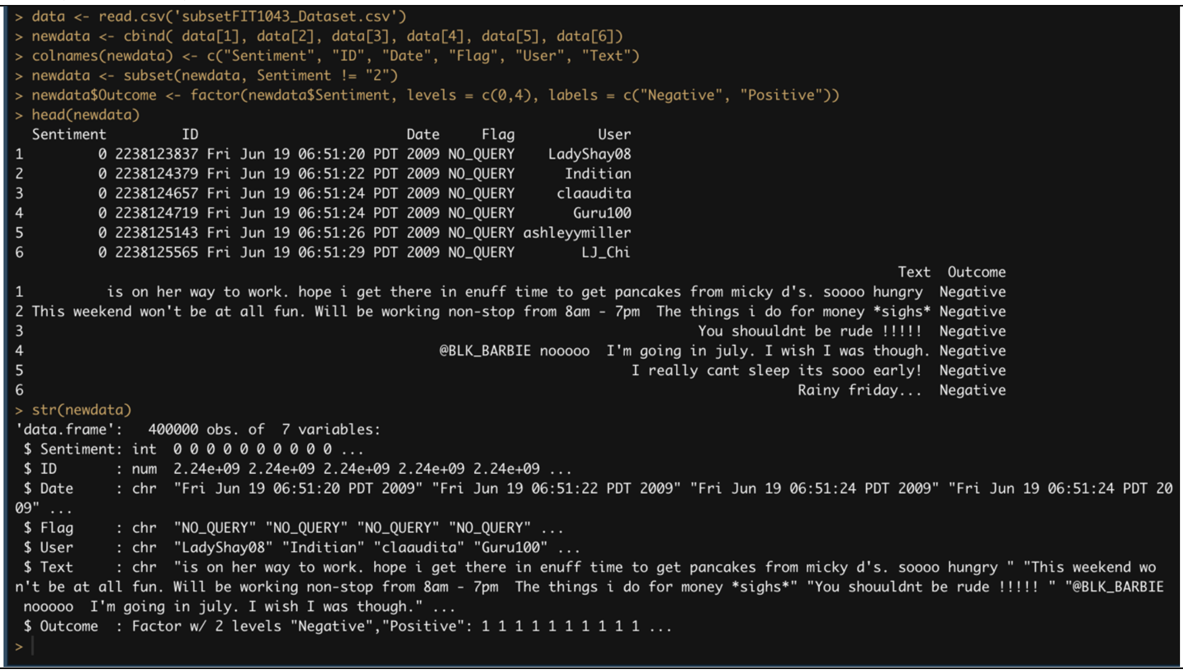
Output 1 Explanation	The output above means that the libraries have been installed and each library is running correcctly.

Explanation 2	This is part 2 of the code. Now that the libraries have been installed, it is time for the real coding. Firstly, I started with the function read.csv to read the csv file that I created earlier and store it inside data as seen in the first line. After that, I noticed there's 6 unnecessary columns created from reading the csv therefore, to remove them I used cbind(data[1], data[2], data[3], data[4], data[5], data[6]) which binds the columns of data 1 to 6 into a new file called new data whilst leaving the rest behind. Now that the data has been cleaned, I added the new column names using the function colnames which changes the names of the columns. It will rename the columns from the first to last. Therefore each "", is one column. After that I created the code subset(newdata, Sentiment != 2) which basically means that in the column sentiment, if there's a row that contains 2 which represents neutral, it won't be counted in the data. A new column has been created called outcome which contains whether the tweet is negative or positive from the help of the column sentiment where 0 means that it's negative and 4 means positive. Now to check whether all the code inputs above has worked I coded head(newdata) to check the first couple lines of the data and str(newdata) to check the different columns. The code is shown below.

Code 2

```
data <- read.csv('subsetFIT1043_Dataset.csv')
newdata <- cbind( data[1], data[2], data[3], data[4], data[5], data[6])
colnames(newdata) <- c("Sentiment", "ID", "Date", "Flag", "User", "Text")
newdata <- subset(newdata, Sentiment != "2")
newdata$Outcome <- factor(newdata$Sentiment, levels = c(0,4), labels =
c("Negative", "Positive"))
head(newdata)
str(newdata)
```

Output 2



```
> data <- read.csv('subsetFIT1043_Dataset.csv')
> newdata <- cbind( data[1], data[2], data[3], data[4], data[5], data[6])
> colnames(newdata) <- c("Sentiment", "ID", "Date", "Flag", "User", "Text")
> newdata <- subset(newdata, Sentiment != "2")
> newdata$Outcome <- factor(newdata$Sentiment, levels = c(0,4), labels = c("Negative", "Positive"))
> head(newdata)
  Sentiment         ID                         Date     Flag          User
1         0 2238123837 Fri Jun 19 06:51:20 PDT 2009 NO_QUERY    LadyShay08
2         0 2238124379 Fri Jun 19 06:51:22 PDT 2009 NO_QUERY      Inditian
3         0 2238124657 Fri Jun 19 06:51:24 PDT 2009 NO_QUERY     claaudita
4         0 2238124719 Fri Jun 19 06:51:24 PDT 2009 NO_QUERY       Guru100
5         0 2238125143 Fri Jun 19 06:51:26 PDT 2009 NO_QUERY ashleyymiller
6         0 2238125565 Fri Jun 19 06:51:29 PDT 2009 NO_QUERY        LJ_Chi
                                                                                              Text  Outcome
1          is on her way to work. hope i get there in enuff time to get pancakes from micky d's. soooo hungry  Negative
2 This weekend won't be at all fun. Will be working non-stop from 8am - 7pm  The things i do for money *sighs* Negative
3                                                                              You shouuldnt be rude !!!!!  Negative
4                                                           @BLK_BARBIE nooooo  I'm going in july. I wish I was though. Negative
5                                                                         I really cant sleep its sooo early!  Negative
6                                                                                          Rainy friday...  Negative
> str(newdata)
'data.frame':	400000 obs. of  7 variables:
 $ Sentiment: int  0 0 0 0 0 0 0 0 0 0 ...
 $ ID       : num  2.24e+09 2.24e+09 2.24e+09 2.24e+09 2.24e+09 ...
 $ Date     : chr  "Fri Jun 19 06:51:20 PDT 2009" "Fri Jun 19 06:51:22 PDT 2009" "Fri Jun 19 06:51:24 PDT 2009" "Fri Jun 19 06:51:24 PDT 20
09" ...
 $ Flag     : chr  "NO_QUERY" "NO_QUERY" "NO_QUERY" "NO_QUERY" ...
 $ User     : chr  "LadyShay08" "Inditian" "claaudita" "Guru100" ...
 $ Text     : chr  "is on her way to work. hope i get there in enuff time to get pancakes from micky d's. soooo hungry " "This weekend wo
n't be at all fun. Will be working non-stop from 8am - 7pm  The things i do for money *sighs*" "You shouuldnt be rude !!!!! " "@BLK_BARBIE
 nooooo  I'm going in july. I wish I was though." ...
 $ Outcome  : Factor w/ 2 levels "Negative","Positive": 1 1 1 1 1 1 1 1 1 1 ...
> 
```

Output 2 Explanation

As seen in the output above, the codes have worked fine and accordingly to plan.

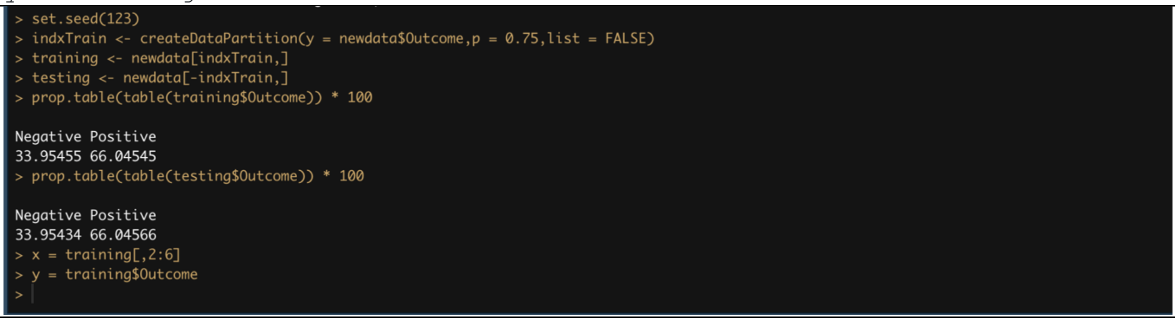
Explanation 3

This is part 3 of the code. For this part it is related to determining the training and testing dataset. Before we determine the training and testing dataset, I used the code set.seed(123) so that I can use this seed for reproducibility. After I used the function createDataPartition which helps create the training to 75% and testing to 25% as seen in p = 0.75 and set y as the Outcome column. After the list has been split, we can use the code newdata[indxTrain,] to store the values into train and for testing it's newdata[-indxTrain,]. After that we can check the values of training and testing by using the code prop.table(table(training$Outcome)) * 100. Lastly to prepare for the naïve bayes algorithm, I set response and feature data x = training [,2:6] and y is the Outcome column. The code is shown below

Code 3

```
set.seed(123)
indxTrain <- createDataPartition(y = newdata$Outcome,p = 0.75,list =
FALSE)
training <- newdata[indxTrain,]
testing <- newdata[-indxTrain,]

prop.table(table(training$Outcome)) * 100
prop.table(table(testing$Outcome)) * 100

x = training[,2:6]
y = training$Outcome
```

Output 3

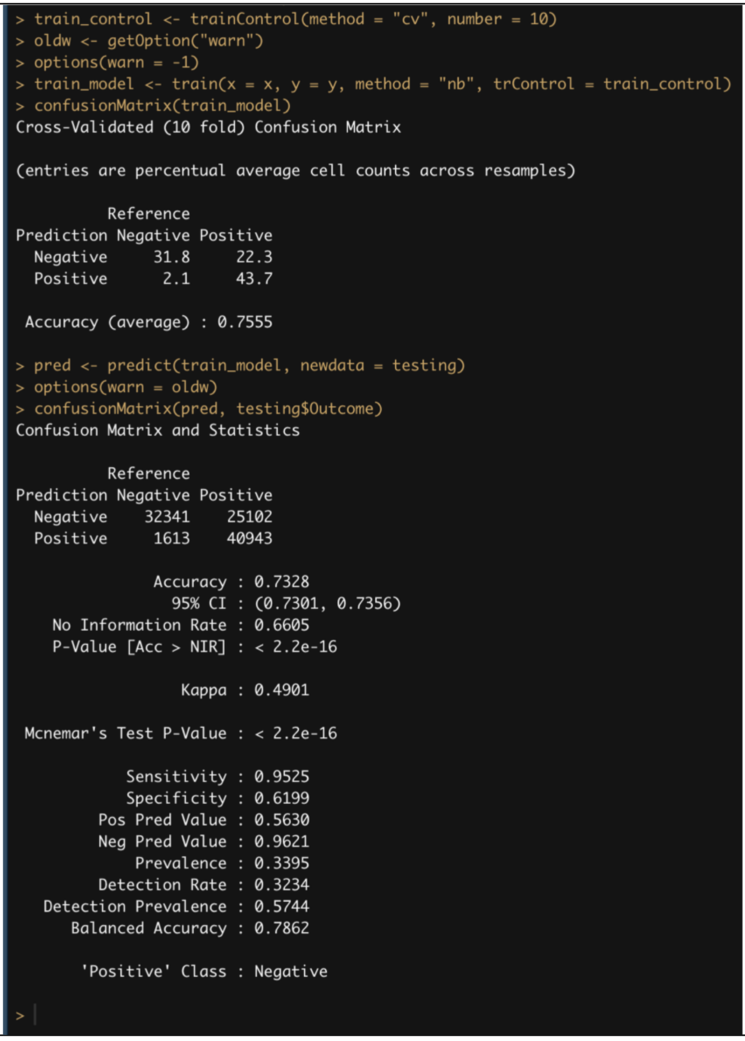
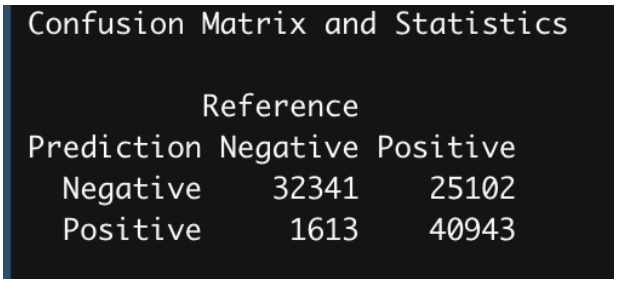


```
> set.seed(123)
> indxTrain <- createDataPartition(y = newdata$Outcome,p = 0.75,list = FALSE)
> training <- newdata[indxTrain,]
> testing <- newdata[-indxTrain,]
> prop.table(table(training$Outcome)) * 100

Negative Positive 
33.95455 66.04545 
> prop.table(table(testing$Outcome)) * 100

Negative Positive 
33.95434 66.04566 
> x = training[,2:6]
> y = training$Outcome
> 
```

Output 3 Explanation	As seen in the output above, the codes have worked fine and accordingly to plan.

Explanation 4	This is the final part of the code where Naïve Bayes is trained, the model tested and lastly the confusion matrix generated. In order to do that, I started with trainControl function which is used to set up 10-fold cross validation procedure then a function called getOption() and options() which is basically used to ignore warnings given in the code. Although I could decide not to use this, and it would be the same however it would look nicer if no warning messages appears when running the code. The warning messages doesn't affect the code much in terms of the output. After the ignore warning functions, the function train() will run where the algorithm Naïve Bayes is trained and then a confusion matrix of the train_model is shown. After that a prediction for testing is ran for the train_model and stored into pred. After that the no warning message code comes to a close by turning back the warn into the old number. This is because it will be bad if the whole code has no warnings therefore that's why I created a temporary no warning code. Lastly the confusion Matrix between the prediction and the outcome column is shown. The codes are below

Code 4	`train_control <- trainControl(method = "cv", number = 10)`
	`oldw <- getOption("warn")`
	`options(warn = -1)`
	
	`train_model <- train(x = x, y = y, method = "nb", trControl = train_control)`
	`confusionMatrix(train_model)`
	
	`pred <- predict(train_model, newdata = testing)`
	
	`options(warn = oldw)`
	`confusionMatrix(pred, testing$Outcome)`

Output 4	
Output 4 Explanation	As seen in the output above, the codes have worked fine and accordingly to plan and also that the accuracy of the naïve bayes algorithm is 73.328% How I understand the confusion Matrix results is like this.  The reference can be changed as the Actual Values and the left prediction can be changed for Predicted Values. The values are also split into 4 different categories which are TP, FP, FN, TN. TP stands for True Positive which means what I've predicted is postive and is true. TP can be found when both the columns colide in Postive, this means the bottom right if you refer to the picture above. The values of True Positive shows 40943 which is the number of I've predicted that's correct. For FP it stands for False positive, this means that what I've predicted is false. This can be found at bottom left where the value is 1613. FP shows the number of predicted values that were postive but actually are negative. FN is False Negative, it means that I've predicted the values to be negative however it turns out to be wrong. FN can be found at the top right corner which has the value of 25102. Lastly, TP stands for True negative which means what I've predicted as negative is true. TP can be found at the top left with a value of 32341. Overall we can see that the number of True postive and True negative is much higher than the false negative and false positive. Although if we add up the numbers of true postive against TP + FP we can see that we get a value of 96.16% precision for postive which is considered very good. However for Negative precision it is only 56.3% which is very low. Overall we can calculate the precision between negative and postive by adding

```
> train_control <- trainControl(method = "cv", number = 10)
> oldw <- getOption("warn")
> options(warn = -1)
> train_model <- train(x = x, y = y, method = "nb", trControl = train_control)
> confusionMatrix(train_model)
Cross-Validated (10 fold) Confusion Matrix

(entries are percentual average cell counts across resamples)

          Reference
Prediction Negative Positive
  Negative     31.8     22.3
  Positive      2.1     43.7

 Accuracy (average) : 0.7555

> pred <- predict(train_model, newdata = testing)
> options(warn = oldw)
> confusionMatrix(pred, testing$Outcome)
Confusion Matrix and Statistics

          Reference
Prediction Negative Positive
  Negative    32341    25102
  Positive     1613    40943

               Accuracy : 0.7328
                 95% CI : (0.7301, 0.7356)
    No Information Rate : 0.6605
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.4901

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9525
            Specificity : 0.6199
         Pos Pred Value : 0.5630
         Neg Pred Value : 0.9621
             Prevalence : 0.3395
         Detection Rate : 0.3234
   Detection Prevalence : 0.5744
      Balanced Accuracy : 0.7862

       'Positive' Class : Negative

> |
```

```
Confusion Matrix and Statistics

          Reference
Prediction Negative Positive
  Negative    32341    25102
  Positive     1613    40943
```

	both presions together and dividing by 200. From that we will get a value of 73.328% which is not very good but also not bad.

Full Code	
	```
	library(rsample)
	library(dplyr)
	library(ggplot2)
	library(caret)
	library(h2o)

	data <- read.csv('subsetFIT1043_Dataset.csv')
	newdata <- cbind( data[1], data[2], data[3], data[4], data[5], data[6])
	colnames(newdata) <- c("Sentiment", "ID", "Date", "Flag", "User", "Text")
	newdata <- subset(newdata, Sentiment != "2")
	newdata$Outcome <- factor(newdata$Sentiment, levels = c(0,4), labels = c("Negative", "Positive"))
	head(newdata)
	str(newdata)

	set.seed(123)
	indxTrain <- createDataPartition(y = newdata$Outcome,p = 0.75,list = FALSE)
	training <- newdata[indxTrain,]
	testing <- newdata[-indxTrain,]

	prop.table(table(training$Outcome)) * 100
	prop.table(table(testing$Outcome)) * 100

	x = training[,2:6]
	y = training$Outcome

	train_control <- trainControl(method = "cv", number = 10)
	oldw <- getOption("warn")
	options(warn = -1)

	train_model <- train(x = x, y = y, method = "nb", trControl = train_control)
	confusionMatrix(train_model)

	pred <- predict(train_model, newdata = testing)

	options(warn = oldw)
	confusionMatrix(pred, testing$Outcome)
	```

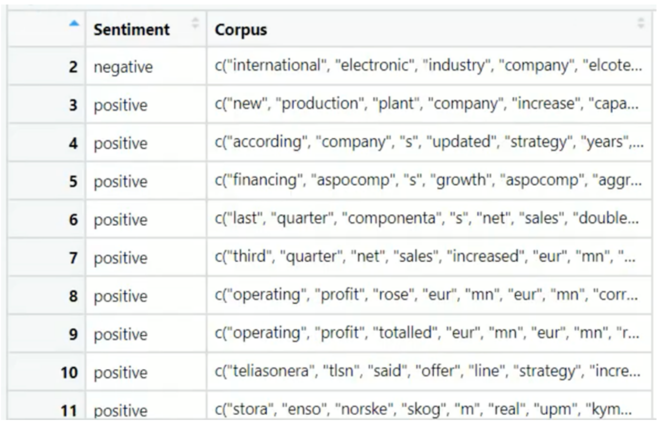
Explanation for text processing	What is text processing? Well text processing in short is sequentially administered macros creation of automated mechanization or modification of electronic text. Here's an example of text processing in naïve algorithm. This code was taken from an external source. Link to the original creator: https://github.com/SpencerPao/Data_Science/blob/main/Naive%20Bayes/Naive_Bayes.R

Code	
	```
	data <- read.csv('all-data.csv', col.names = c("Sentiment", "Corpus"))
	summary(data)
	str(data)
	table(data$Sentiment)

	data <- subset(data, Sentiment != "neutral")
	table(data$Sentiment)
	```

```r
init_neg <- (table(data$Sentiment) / sum(table(data$Sentiment)))[1]
init_pos <- (table(data$Sentiment) / sum(table(data$Sentiment)))[2]

library(stopwords)
library(tokenizers)
library(SnowballC)
library(dplyr)

data$Corpus <- (tokenize_words(data$Corpus,
                               lowercase = TRUE,
                               strip_punct = TRUE,
                               strip_numeric = TRUE,
                               stopwords = stopwords::stopwords("en")))

calc_Probs <- function(tokens) {
  counts <- table(unlist(tokens)) + 1
  (counts/sum(counts))
}

pos_data <- subset(data, Sentiment == "positive")
neg_data <- subset(data, Sentiment == "negative")

pos_probs <- calc_Probs(pos_data$Corpus)
neg_probs <- calc_Probs(neg_data$Corpus)

classify_sentiment <- function(input) {
  test <- unlist(tokenize_words(input,
                         lowercase = TRUE,
                         strip_punct = TRUE,
                         strip_numeric = TRUE,
                         stopwords = stopwords::stopwords("en")))
  print(test)
  pos_pred <- init_pos * ifelse(is.na(prod(pos_probs[test])), 1,
prod(pos_probs[test]))
  neg_pred <- init_neg * ifelse(is.na(prod(neg_probs[test])), 1,
prod(neg_probs[test]))
  if (pos_pred > neg_pred){
    return ("Positive")
  } else {
    return ("Negative")
  }
}
```

Explanation of the code	This code is for Naïve Bayes Algorithm. What do I understand from the code above here? Well like how my Naïve Bayes Algorithm worked, I started with read csv functions to read the csv and he also removed the rows that contained neutral. So far everything implemented for this Naïve Bayes algorithm is like what I've coded. However, the biggest difference is at line 12 where he started using libraries which include tokenizers which is a commonly used library in r programming for text processing. What tokenizing normally does is covert a long string into individual words. From there onwards the person implemented the function tokenize_words() which is used to clean the text words from column corpus as he has stated. Also, he choose lowercase = TRUE which means that after the cleaning process is completed, all the words will be turned into lower case. Another code he implemented into the function tokenize_words is strip_punct, strip_numeric and stopwords. What strip_punct do is remove all the punctuation from the long string, for strip_numeric it removes all numeric numbers and lastly stopwords is to remove redundant words which are useless and follows the language en which is English.

After the use of the function tokenize_words() the output of the new table should look like



	Sentiment	Corpus
2	negative	c("international", "electronic", "industry", "company", "elcote...
3	positive	c("new", "production", "plant", "company", "increase", "capa...
4	positive	c("according", "company", "s", "updated", "strategy", "years",...
5	positive	c("financing", "aspocomp", "s", "growth", "aspocomp", "aggr...
6	positive	c("last", "quarter", "componenta", "s", "net", "sales", "double...
7	positive	c("third", "quarter", "net", "sales", "increased", "eur", "mn", "...
8	positive	c("operating", "profit", "rose", "eur", "mn", "eur", "mn", "corr...
9	positive	c("operating", "profit", "totalled", "eur", "mn", "eur", "mn", "r...
10	positive	c("teliasonera", "tlsn", "said", "offer", "line", "strategy", "incre...
11	positive	c("stora", "enso", "norske", "skog", "m", "real", "upm", "kym...

As we can see from the csv data file now, in the corpus column which originally contained long strings has been tokenized and turned into list of words which could be used later.

For the next code, he's used a helper function to calculate the frequency of the words in the given class. After that he split the sentiment into it's respective categories of Negative and positive. Then he calculates the probabilities through the earlier created function on the column data corpus.

Now the earlier tokenize function will be repeated again but this time it will be used for the testing later where again the word will go through the function tokenize_words then all the in function code like lowercase then it will print out the test after being tokenized. Then there's a code which calculates the prediction of whether the predicted value is a positive or negative.

That's all for his code. Basically what I have learned is that text processing works through looking at a whole long string of words where it will be processed through the use of tokenize function and some more code within the function to get a nicely processed text where each word has been separated and processed.