MONASH University

FIT5145: Foundations of Data Science

Assignment-4

Bhavna Balakrishnan

33954437

TASK-A

1. Range for sold time: For this question I will be writing code to return to us the earliest and latest sold time to understand the range better

CODE:

```
Last login: Wed May 29 11:57:42 on ttys001
% awk -F',' 'NR==1 || ($4 != "" && $4 != "NA")'
"/Users/bhavna_balakrishnan/Desktop/Uni/Foundations
DS/ass4/property_transaction_victoria.csv" | iconv -f ISO-8859-1 -t
UTF-8 > cleaned_property_transaction.csv
% tail -n +2 cleaned_property_transaction.csv >
cleaned_no_header.csv
% LC_ALL=C sort -t ',' -k4,4 cleaned_no_header.csv >
sorted_property_transaction.csv
% head -n 1 sorted_property_transaction.csv | awk -F',' '{print
"Earliest sold_time: "$4}'
Earliest sold_time: 1/01/2021 0:19
% tail -n 1 sorted_property_transaction.csv | awk -F',' '{print
"Latest sold_time: "$4}'
Latest sold_time: 9/12/2021 9:58
```

ANSWER:

As we can see highlighted in the above code we have returned the results for earliest sold time as 01/01/2021 at 0:19 and the latest on 9/12/21 at 9:58.

EXPLANATION

I have done the above code step by step (we can see by the % symbol)

Step 1: Cleaning the data and removing the NA and empty values

```
awk -F',' 'NR==1 || ($4 != "" && $4 != "NA")'
"/Users/bhavna_balakrishnan/Desktop/Uni/Foundations
DS/ass4/property_transaction_victoria.csv" | iconv -f ISO-8859-1 -t UTF-8 >
cleaned_property_transaction.csv
```

•awk -F',' this command is asking the terminal to recognise commas ‘,’ as a delimiter.
•'NR==1 || ($4 != "" && $4 != "NA") This line goes line by line through the data using logical OR operators to check for null or NA values. It then ensures the values we are looking to return are neither empty or NA value.
•"/Users/bhavna_balakrishnan/Desktop/Uni/Foundations DS/ass4/property_transaction_victoria.csv : This is just us feeding the address of the file.
•iconv -f ISO-8859-1 -t UTF-8 > cleaned_property_transaction.csv: We convert format to UTF-8.

Step-2: Removing the header

•tail -n +2 cleaned_property_transaction.csv > cleaned_no_header.csv: This line essentially removes the header from the processing so that it will return only the date-time values

Step-3: Sorting data and returning results

•LC_ALL=C sort -t ',' -k4,4 cleaned_no_header.csv > sorted_property_transaction.csv: This line sorts the data in order.
•head -n 1 sorted_property_transaction.csv | awk -F',' '{print "Earliest sold_time: "$4}': This line returns to us the ealiest sold time (col 4 is the sold time one)
•tail -n 1 sorted_property_transaction.csv | awk -F',' '{print "Latest sold_time: "$4}': This line returns the latest sold time.

2. IDs and sold times

    a. Non-numeric values and length more than 6

CODE:

```
% awk -F',' 'NR > 1 && ($1 !~ /^[0-9]{6}$/)'
"/Users/bhavna_balakrishnan/Desktop/Uni/Foundations
DS/ass4/property_transaction_victoria.csv" | wc -l
```

```
10
```

ANSWER:

We can see that the result returned to us as highlighted above is that 10 fields do not contain numeric values or have length more than 6.

EXPLANATION:

□ Like the previous code, awk -F',' is marking commas as a delimiter.
□ NR>1 means we are not taking the header into consideration.
□ ($1 !~ /^[0-9]{6}$/)is the code which checks column 1 is the ID column that the code checks in and the rest is a condition to check for non-numeric characters more than 6 digits.
□ wc -l pipes out the output.

    b. Removing filtered values and Sold time part from preceding task

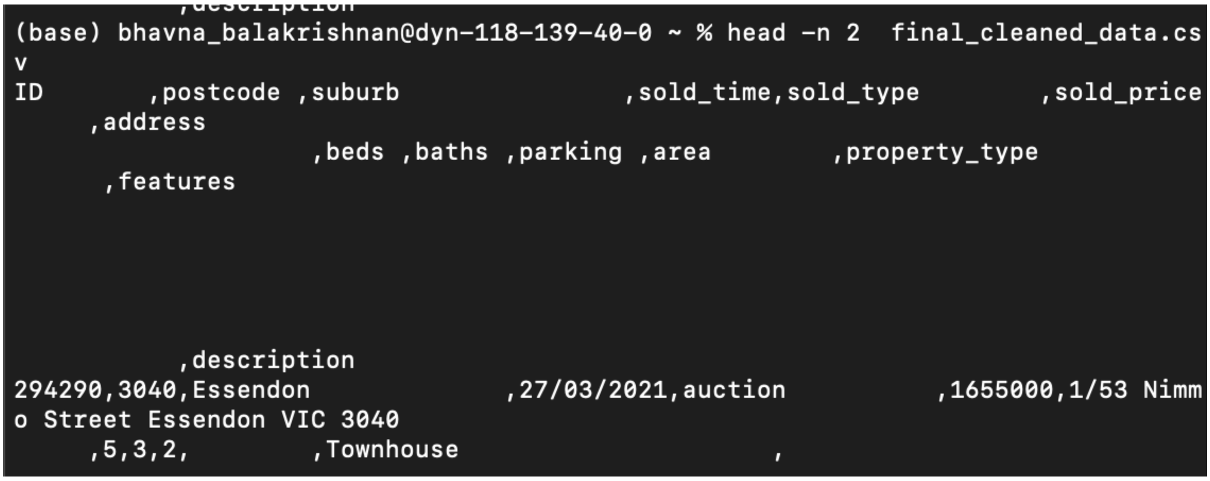
CODE:

```
% awk -F',' 'NR==1 || ($1 ~ /^[0-9]{6}$/)'
"/Users/bhavna_balakrishnan/Desktop/Uni/Foundations
DS/ass4/property_transaction_victoria.csv" > cleaned_ids.csv
% awk -F',' 'BEGIN {OFS=","} {split($4, a, " "); $4=a[1]; print}'
cleaned_ids.csv > final_cleaned_data.csv
```

Testing line

```
% head -n 2  final_cleaned_data.csv
```

OUTPUT:



```
              ,description
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ % head -n 2  final_cleaned_data.cs
v
ID        ,postcode ,suburb               ,sold_time,sold_type        ,sold_price
     ,address
                     ,beds ,baths ,parking ,area        ,property_type
      ,features






            ,description
294290,3040,Essendon            ,27/03/2021,auction          ,1655000,1/53 Nimm
o Street Essendon VIC 3040
     ,5,3,2,        ,Townhouse                    ,
```

EXPLANATION:

As explained in the previous task, the identified values that were non numeric or had longer than 6-digit IDs were identified. Then the rest of the data was redirected to a new path named cleaned_ids.csv after filtering out the mentioned values.

- □ 'BEGIN {OFS=","} this command is executed prior to any processing of input lines.
- □ {split($4, a, " "); $4=a[1] this is the main command that will remove the time from the sold_date column 4 is sold date and a[1] means it wants us to return the first value which is just the date by splitting it.
- □ Finally, I also used a head -n 2 command to see what output I was getting to ensure that the sold_time showed me only the date and not the time.

c. First 5 lines and filtered_property.csv

CODE:

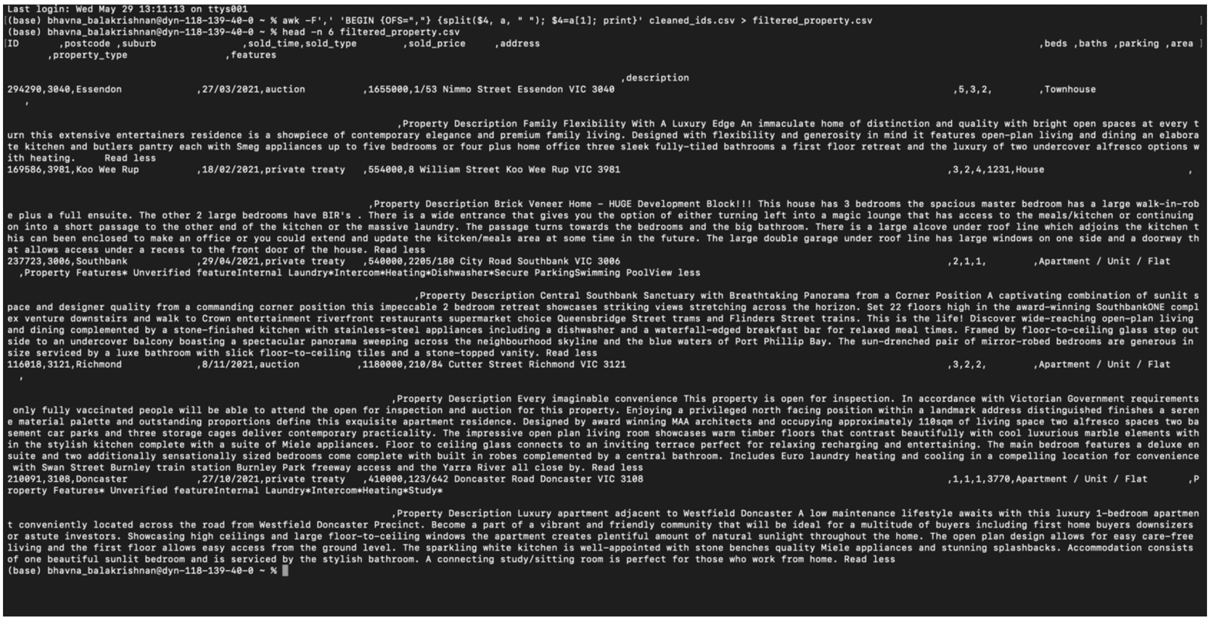
```
% awk -F',' 'NR==1 || ($1 ~ /^[0-9]{6}$/)'
"/Users/bhavna_balakrishnan/Desktop/Uni/Foundations
DS/ass4/property_transaction_victoria.csv" > cleaned_ids.csv
% awk -F',' 'BEGIN {OFS=","} {split($4, a, " "); $4=a[1]; print}'
cleaned_ids.csv > filtered_property.csv
% head -n 6 filtered_property.csv
```

ANSWER

Below we can see the output of the above command-

```
ID        ,postcode ,suburb               ,sold_time,sold_type        ,sold_price     ,address
,beds ,baths ,parking ,area        ,property_type                 ,features
,description
294290,3040,Essendon             ,27/03/2021,auction          ,1655000,1/53 Nimmo Street
Essendon VIC 3040                                                            ,5,3,2,
,Townhouse                    ,
,Property Description Family Flexibility With A Luxury Edge An immaculate home of distinction
and quality with bright open spaces at every turn this extensive entertainers residence is a
showpiece of contemporary elegance and premium family living. Designed with flexibility and
generosity in mind it features open-plan living and dining an elaborate kitchen and butlers
pantry each with Smeg appliances up to five bedrooms or four plus home office three sleek
fully-tiled bathrooms a first floor retreat and the luxury of two undercover alfresco options
with heating.     Read less
169586,3981,Koo Wee Rup          ,18/02/2021,private treaty   ,554000,8 William Street Koo
Wee Rup VIC 3981                                                        ,3,2,4,1231,House
,
,Property Description Brick Veneer Home - HUGE Development Block!!! This house has 3 bedrooms
```

```
the spacious master bedroom has a large walk-in-robe plus a full ensuite. The other 2 large
bedrooms have BIR's . There is a wide entrance that gives you the option of either turning
left into a magic lounge that has access to the meals/kitchen or continuing on into a short
passage to the other end of the kitchen or the massive laundry. The passage turns towards the
bedrooms and the big bathroom. There is a large alcove under roof line which adjoins the
kitchen this can been enclosed to make an office or you could extend and update the
kitcken/meals area at some time in the future. The large double garage under roof line has
large windows on one side and a doorway that allows access under a recess to the front door
of the house. Read less
237723,3006,Southbank            ,29/04/2021,private treaty   ,540000,2205/180 City Road
Southbank VIC 3006                                                         ,2,1,1,
,Apartment / Unit / Flat       ,Property Features* Unverified featureInternal
Laundry*Intercom*Heating*Dishwasher*Secure ParkingSwimming PoolView less
,Property Description Central Southbank Sanctuary with Breathtaking Panorama from a Corner
Position A captivating combination of sunlit space and designer quality from a commanding
corner position this impeccable 2 bedroom retreat showcases striking views stretching across
the horizon. Set 22 floors high in the award-winning SouthbankONE complex venture downstairs
and walk to Crown entertainment riverfront restaurants supermarket choice Queensbridge Street
trams and Flinders Street trains. This is the life! Discover wide-reaching open-plan living
and dining complemented by a stone-finished kitchen with stainless-steel appliances including
a dishwasher and a waterfall-edged breakfast bar for relaxed meal times. Framed by floor-to-
ceiling glass step outside to an undercover balcony boasting a spectacular panorama sweeping
across the neighbourhood skyline and the blue waters of Port Phillip Bay. The sun-drenched
pair of mirror-robed bedrooms are generous in size serviced by a luxe bathroom with slick
floor-to-ceiling tiles and a stone-topped vanity. Read less
116018,3121,Richmond             ,8/11/2021,auction          ,1180000,210/84 Cutter Street
Richmond VIC 3121                                                        ,3,2,2,
,Apartment / Unit / Flat       ,
,Property Description Every imaginable convenience This property is open for inspection. In
accordance with Victorian Government requirements only fully vaccinated people will be able
to attend the open for inspection and auction for this property. Enjoying a privileged north
facing position within a landmark address distinguished finishes a serene material palette and
outstanding proportions define this exquisite apartment residence. Designed by award winning
MAA architects and occupying approximately 110sqm of living space two alfresco spaces two
basement car parks and three storage cages deliver contemporary practicality. The impressive
open plan living room showcases warm timber floors that contrast beautifully with cool
luxurious marble elements within the stylish kitchen complete with a suite of Miele
appliances. Floor to ceiling glass connects to an inviting terrace perfect for relaxing
recharging and entertaining. The main bedroom features a deluxe ensuite and two additionally
sensationally sized bedrooms come complete with built in robes complemented by a central
bathroom. Includes Euro laundry heating and cooling in a compelling location for convenience
with Swan Street Burnley train station Burnley Park freeway access and the Yarra River all
close by. Read less
210091,3108,Doncaster            ,27/10/2021,private treaty   ,410000,123/642 Doncaster Road
Doncaster VIC 3108                                                     ,1,1,1,3770,Apartment
/ Unit / Flat       ,Property Features* Unverified featureInternal
Laundry*Intercom*Heating*Study*
,Property Description Luxury apartment adjacent to Westfield Doncaster A low maintenance
lifestyle awaits with this luxury 1-bedroom apartment conveniently located across the road
from Westfield Doncaster Precinct. Become a part of a vibrant and friendly community that will
be ideal for a multitude of buyers including first home buyers downsizers or astute investors.
Showcasing high ceilings and large floor-to-ceiling windows the apartment creates plentiful
amount of natural sunlight throughout the home. The open plan design allows for easy care-
free living and the first floor allows easy access from the ground level. The sparkling white
kitchen is well-appointed with stone benches quality Miele appliances and stunning
splashbacks. Accommodation consists of one beautiful sunlit bedroom and is serviced by the
stylish bathroom. A connecting study/sitting room is perfect for those who work from home.
Read less
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ %
```



```
Last login: Wed May 29 13:11:13 on ttys001
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ % awk -F',' 'BEGIN {OFS=","} {split($4, a, " "); $4=a[1]; print}' cleaned_ids.csv > filtered_property.csv
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ % head -n 6 filtered_property.csv
ID       ,postcode ,suburb               ,sold_time,sold_type        ,sold_price     ,address                                                                                            ,beds ,baths ,parking ,area
       ,property_type                  ,features

                                                                                                                                   ,description
294290,3040,Essendon             ,27/03/2021,auction          ,1655000,1/53 Nimmo Street Essendon VIC 3040                                                                                   ,5,3,2,        ,Townhouse
   ,

                                                                                              ,Property Description Family Flexibility With A Luxury Edge An immaculate home of distinction and quality with bright open spaces at every t
urn this extensive entertainers residence is a showpiece of contemporary elegance and premium family living. Designed with flexibility and generosity in mind it features open-plan living and dining an elabora
te kitchen and butlers pantry each with Smeg appliances up to five bedrooms or four plus home office three sleek fully-tiled bathrooms a first floor retreat and the luxury of two undercover alfresco options w
ith heating.     Read less
169586,3981,Koo Wee Rup          ,18/02/2021,private treaty   ,554000,8 William Street Koo Wee Rup VIC 3981                                                                                  ,3,2,4,1231,House              ,


                                                                                         ,Property Description Brick Veneer Home - HUGE Development Block!!! This house has 3 bedrooms the spacious master bedroom has a large walk-in-rob
e plus a full ensuite. The other 2 large bedrooms have BIR's . There is a wide entrance that gives you the option of either turning left into a magic lounge that has access to the meals/kitchen or continuing
on into a short passage to the other end of the kitchen or the massive laundry. The passage turns towards the bedrooms and the big bathroom. There is a large alcove under roof line which adjoins the kitchen t
his can been enclosed to make an office or you could extend and update the kitcken/meals area at some time in the future. The large double garage under roof line has large windows on one side and a doorway th
at allows access under a recess to the front door of the house. Read less
237723,3006,Southbank            ,29/04/2021,private treaty   ,540000,2205/180 City Road Southbank VIC 3006                                                                                  ,2,1,1,        ,Apartment / Unit / Flat
  ,Property Features* Unverified featureInternal Laundry*Intercom*Heating*Dishwasher*Secure ParkingSwimming PoolView less

                                                                                                  ,Property Description Central Southbank Sanctuary with Breathtaking Panorama from a Corner Position A captivating combination of sunlit s
pace and designer quality from a commanding corner position this impeccable 2 bedroom retreat showcases striking views stretching across the horizon. Set 22 floors high in the award-winning SouthbankONE compl
ex venture downstairs and walk to Crown entertainment riverfront restaurants supermarket choice Queensbridge Street trams and Flinders Street trains. This is the life! Discover wide-reaching open-plan living
and dining complemented by a stone-finished kitchen with stainless-steel appliances including a dishwasher and a waterfall-edged breakfast bar for relaxed meal times. Framed by floor-to-ceiling glass step out
side to an undercover balcony boasting a spectacular panorama sweeping across the neighbourhood skyline and the blue waters of Port Phillip Bay. The sun-drenched pair of mirror-robed bedrooms are generous in
size serviced by a luxe bathroom with slick floor-to-ceiling tiles and a stone-topped vanity. Read less
116018,3121,Richmond             ,8/11/2021,auction          ,1180000,210/84 Cutter Street Richmond VIC 3121                                                                                 ,3,2,2,        ,Apartment / Unit / Flat
  ,

                                                                                             ,Property Description Every imaginable convenience This property is open for inspection. In accordance with Victorian Government requirements
 only fully vaccinated people will be able to attend the open for inspection and auction for this property. Enjoying a privileged north facing position within a landmark address distinguished finishes a seren
e material palette and outstanding proportions define this exquisite apartment residence. Designed by award winning MAA architects and occupying approximately 110sqm of living space two alfresco spaces two ba
sement car parks and three storage cages deliver contemporary practicality. The impressive open plan living room showcases warm timber floors that contrast beautifully with cool luxurious marble elements with
in the stylish kitchen complete with a suite of Miele appliances. Floor to ceiling glass connects to an inviting terrace perfect for relaxing recharging and entertaining. The main bedroom features a deluxe en
suite and two additionally sensationally sized bedrooms come complete with built in robes complemented by a central bathroom. Includes Euro laundry heating and cooling in a compelling location for convenience
 with Swan Street Burnley train station Burnley Park freeway access and the Yarra River all close by. Read less
210091,3108,Doncaster            ,27/10/2021,private treaty   ,410000,123/642 Doncaster Road Doncaster VIC 3108                                                                              ,1,1,1,3770,Apartment / Unit / Flat       ,P
roperty Features* Unverified featureInternal Laundry*Intercom*Heating*Study*

                                                                                             ,Property Description Luxury apartment adjacent to Westfield Doncaster A low maintenance lifestyle awaits with this luxury 1-bedroom apartmen
t conveniently located across the road from Westfield Doncaster Precinct. Become a part of a vibrant and friendly community that will be ideal for a multitude of buyers including first home buyers downsizers
or astute investors. Showcasing high ceilings and large floor-to-ceiling windows the apartment creates plentiful amount of natural sunlight throughout the home. The open plan design allows for easy care-free
living and the first floor allows easy access from the ground level. The sparkling white kitchen is well-appointed with stone benches quality Miele appliances and stunning splashbacks. Accommodation consists
of one beautiful sunlit bedroom and is serviced by the stylish bathroom. A connecting study/sitting room is perfect for those who work from home. Read less
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ %
```

EXPLANATION:
Here I have run a similar code to the previous task with the only two differences-

- Renaming cleaned.ids.csv> filtered_property.csv per requirement
- Using head -n 6 since first result would return the header.

3. Mount Dandenong

CODE

```
% iconv -f ISO-8859-1 -t UTF-8 filtered_property.csv >
filtered_property_utf8.csv
% awk -F',' 'NR==1 || $7 ~ /Mount Dandenong/'
filtered_property_utf8.csv > mount_dandenong_filtered.csv
% LC_ALL=C sort -t ',' -k4,4 mount_dandenong_filtered.csv >
sorted_mount_dandenong.csv
% echo "Earliest mention of 'Mount Dandenong':"
head -n 2 sorted_mount_dandenong.csv | tail -n 2
echo “Latest mention of ‘Mount Dandenong’:”
tail -n 2 sorted_mount_dandenong.csv
```

ANSWER

```
Earliest mention of 'Mount Dandenong':
127976,3767,Mount Dandenong      ,1/01/2021,private treaty   ,765000,16 Oakley Street Mount
Dandenong VIC 3767                                                     ,3,1,1,1304,House
,
,Property Description Traditional Space with a Magnificent Modern Renovation | $755000 - $780000
Perfectly perched up in an elevated and picturesque position this beautifully updated home enjoys a
sprawling landscape and traditional floor plan that boasts light-filled living zones modern tones and
exquisite contemporary finishes. Manicured hedges are a welcoming feature before you step inside
onto warming wooden floorboards that make their way through a lounge and dining area equipped with a
fire place and split-system air-conditioning for comfort.       Read less
199939,3767,Mount Dandenong      ,1/10/2021,private treaty   ,1000000,1496 Mount Dandenong Tourist
Road Mount Dandenong VIC 3767                                   ,4,3,6,1911,House
,Property Features* Unverified featureGas*Broadband internet access*Fireplace(s)*Dishwasher*Fully
fenced*DeckView less
,Property Description Best investment potential in the hills !!! Situated on 1910m2 (approx.) of
flat fully fenced land is this great property that offers an investors heaven! Offering 3 separate
accommodation options each fully self contained. The hills cottage and dual studio units alongside
all ooze character and have their own verandas set behind rhododendrons and camellias. Freshly
painted and renovated throughout the cottage and 2 units within walking distance to the shops and
all the amenities of Mount Dandenong and Olinda villages with this property's outstanding location
```

```
and charm offer a rare combination. Main Cottage: The fully renovated main cottage circa 1947
offers 2 bedroom and recently updated bathroom. With a traditional floor plan with high ceilings a
spacious open plan lounge and dining room with wood fire encased within an original brick fireplace
plenty of room to entertain. Large original windows and Baltic floors lead down the main hall to the
country style kitchen and meals area. Country style kitchen with new benchtops cabinets splashback
sink tap fittings gas oven/cooktop rangehood and dishwasher with spacious adjoining laundry/utility
room leading out to the garden. Additional Features: Freshly painted throughout new blinds split
system Electric hot water service Electric wall panel heaters Energy-efficient LED lighting NBN
connection. Cottage Size: Approx 100m2 Potential Annual Rent: $24960 @ $480 per week (including
water usage) Outdoors the established European styled gardens will offer color in Spring scattered
along the long driveway. A veggie patch and herb garden plus fire pit area out the back of the
cottage with the added benefit of plenty of off street parking plus entertaining area.... this
property is full of possibilities. Rear Studio Unit: The rear studio unit Unit 2 has been freshly
updated perfect for a new tenant or multi generational family options. Freshly painted and updated
throughout offering one bedroom one recently updated bathroom with open plan lounge kitchenette
featuring a gas oven/cooktop dishwasher electric hot water electric wall panel heaters new blinds.
Approx. Approx. Studio Size: 35m2 Potential Annual Rent: $18250 @ approx. $350 per week (includes
water) Unit 2 is currently vacant so the options are yours! Shared laundry facilities between the
two units allow for additional privacy. Front Studio Unit: Front studio unit Unit 3 is currently
tenanted til March 2022. Freshly painted and updated throughout. Offering one bedroom one bathroom
with open plan lounge kitchenette gas oven/cooktop electric hot water electric wall panel heaters
new blinds and a great covered front porch. Approx. Studio Size: 42m2 Potential Annual Rent: $17160
@ $330 per week (includes water) Property has been refreshed repainted and refurbished including
electrical and plumbing ready for the next twenty years of tenants with a great recurring annual
gross rental income from the three tenancies close to $60000 + CPI. For the investor this property
yields a gross rental return of approx. 6%. Contact us via email for further details or to book a
time to inspect privately. Please note: All property details shown are correct at time of
publishing. Some properties may have been sold in the preceding 24 hours and we recommend that you
confirm open for inspection times with the listing agent direct or the listing office. Read less

“Latest mention of ‘Mount Dandenong’:”
198606,3787,Sassafras            ,8/09/2021,private treaty   ,890000,338 Mount Dandenong Tourist
Road Sassafras VIC 3787                                            ,3,1,1,2268,House
,Property Features* Unverified featureBath*Garden*BalconyBuilt in wardrobesDishwasherFloorboardsGas
Ducted HeatingWood HeaterView less
,"Property Description Serenity in Sassafras **ZOOM VIRTUAL TOUR LINK AVAILABLE ON REQUEST** Wake
each morning to a blissful sunrise or cascading fog through the surrounding bush setting. Make the
ambiance of living in a tree house with nature cocooning you a part of your everyday living! Only a
brief walk to the Sassafras village come and be a part of this amazing community with a beautiful
communal vibe. The home is all about the natural environment. Set over two levels and offering 3
bedrooms each with built-in robes and a central bathroom the home is warm and inviting. Imagine a
fire crackling whist there are snow predictions. This all adds to the daily winter pleasure.
Alternatively there are evening sunsets behind us imagine sitting on the balcony watching the
birdlife in the garden and trees. Nature abounds with Kookaburras King Parrots and Rosellas
together with visits from wallabies and echidnas in the natural bush environment. ""Sassy Lane"" as
neighbours refer to it is a special place of a close-knit group supporting each other. Every part
of this home is beautifully presented for a relaxed indoor-outdoor lifestyle that works in harmony
with its natural surroundings. Set on an easy-care natural bush half-acre (2268 m2 approx.)
property.  Read less"
```

EXPLANATION

- The first line iconv -f ISO-8859-1 -t UTF-8 filtered_property.csv > filtered_property_utf8.csv basically transforms the format to UTF 8 for more readability.
- We then employ another comma delimiter followed by defining the column in command 'NR==1 || $7 ~/Mount Dandenong/ uses a logical OR Operator which checks to see if the condition is true
- Then the filtered and converted file gets sorted and we use echo head and tail commands to return our results.

4. Suburbs
   a.
   ```
   % awk -F',' 'NR > 1 {print $3}' filtered_property.csv | sort
   | uniq | wc -l
       1410
   ```
   b.
   ```
   % awk -F',' 'NR > 1 {print $3}' filtered_property.csv | sort
   | uniq -c | sort -nr | head -n 5
   ```

```
965 Melbourne
935 Reservoir
917 Frankston
915 Berwick
892 Pakenham
```

ANSWER:

As we can see above, the number of uniques suburbs in the filtered dataset is 1410 and the top 5 suburbs are Melbourne, Reservoir, Frankston, Berwick and Pakenham.

EXPLANATION:

a. As we can see from the only line of code, it is basically again establishing a comma delimiter and defining the search column number 3 suburbs and sorting as well as looking for unique values.

b. In this part, the sorted data is taken and we ask to return top 5 results.

5. Description

    a. Alfresco and Renovation

CODE

```
bhavna_balakrishnan@dyn-118-139-40-0 ~ % awk -F',' 'NR > 1 &&
tolower($14) ~ /alfresco/ && tolower($14) ~ /renovation/'
filtered_property.csv | wc -l
```

ANSWER

```
548
```

There are 548 entries that mention ‘Alfresco’ and ‘Renovation’ in their description.

EXPLANATION

□ Again we use delimiters and skip the header.

□ Tolower($14) this command convers the 14th column ‘description’ into lowercase.

□ By stating ~/alfresco/ and ~/renovation/ we are checking to see if the words are present in the description

    b. Property Size

CODE

```
~ % awk -F',' 'NR > 1 && tolower($14) ~ /[0-9]+[ ]?(m2|sq[
]?metres)/' filtered_property.csv | wc -l
```

ANSWER

```
21329
```

There are 21329 entries with descriptions mentioning property size in them.

EXPLANATION:

This command also exempts the header, gives comma as delimiter and converts the description column contents into lower case. It then looks for property size descriptions like m2 and sq meters by using logical OR operators and regular expressions.

6. Columns and conditions
    a. Columns

CODE:

```
% awk -F',' 'BEGIN {OFS=","} {print $1, $4, $5, $6, $7, $8, $11,
$12, $14}' filtered_property.csv > step1_filtered.csv
```

EXPLANATION:
Here this code is prompting only to print out required columns numbered 1,4,5,6,7,8,11,12 and 14. Then it filters this and establishes a new dataset step1_filtered.csv

    b. Conditions
        i. Odd Months

CODE:

```
% awk -F',' 'BEGIN {OFS=","} NR==1 {print; next} {
split($2, date, "/");
month = date[2];
if (month % 2 == 1) print
}' step1_filtered.csv > step2_filtered.csv
```

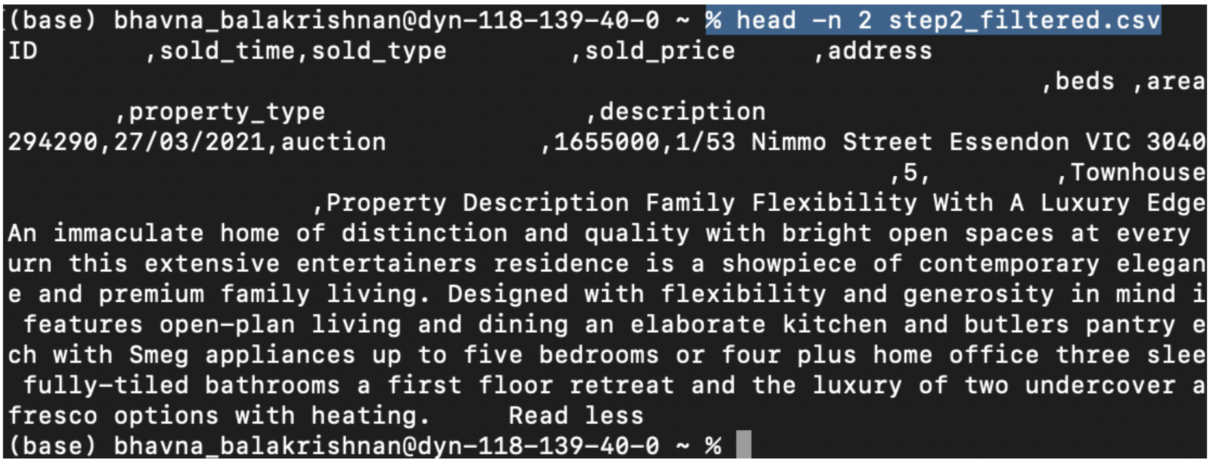
EXPLANATION:

This command does the standard removal of header and establishment of comma as delimiter then gives logic to compute odd months by checking remainder =1 when divided by 2. From here it further filters data to step2_filtered data.

TESTING:

```
% head -n 2 step2_filtered.csv
```

Running this command to test above code works



```
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ % head -n 2 step2_filtered.csv
ID       ,sold_time,sold_type        ,sold_price     ,address
                                                                    ,beds  ,area
       ,property_type                 ,description
294290,27/03/2021,auction          ,1655000,1/53 Nimmo Street Essendon VIC 3040
                                                          ,5,        ,Townhouse
                    ,Property Description Family Flexibility With A Luxury Edge
An immaculate home of distinction and quality with bright open spaces at every
urn this extensive entertainers residence is a showpiece of contemporary elegan
e and premium family living. Designed with flexibility and generosity in mind i
 features open-plan living and dining an elaborate kitchen and butlers pantry e
ch with Smeg appliances up to five bedrooms or four plus home office three slee
 fully-tiled bathrooms a first floor retreat and the luxury of two undercover a
fresco options with heating.     Read less
(base) bhavna_balakrishnan@dyn-118-139-40-0 ~ %
```

        ii. Townhouse

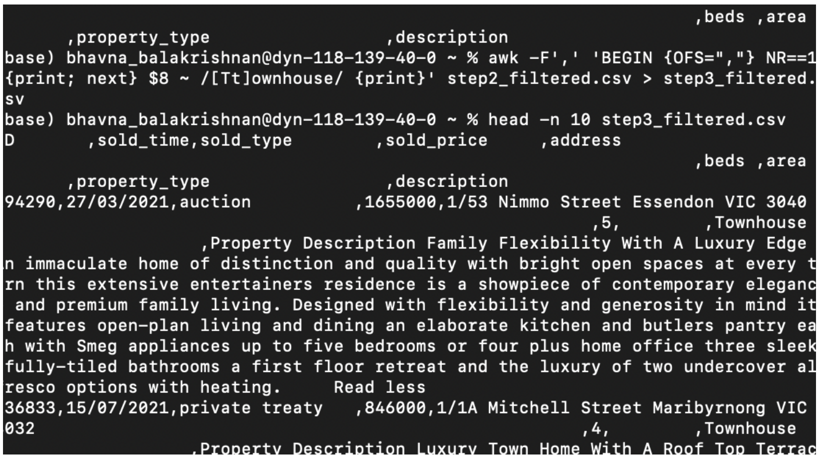
CODE:

```
% awk -F',' 'BEGIN {OFS=","} NR==1 {print; next} $8 ~ /[Tt]ownhouse/
{print}' step2_filtered.csv > step3_filtered.csv
```

EXPLANATION:

Here this command alos uses OFS=',' to ensure that the output field separator is a comma and it maintains the format of CSV.Here the new column styles see that column 8 is property type which we set condition to serach for 'Townhouse'.Finally we save this step as step3_filtered.csv.

I also used a -n 10 header command to test this step worked-



```
                                                                        ,beds ,area
      ,property_type                 ,description
base) bhavna_balakrishnan@dyn-118-139-40-0 ~ % awk -F',' 'BEGIN {OFS=","} NR==1
{print; next} $8 ~ /[Tt]ownhouse/ {print}' step2_filtered.csv > step3_filtered.
sv
base) bhavna_balakrishnan@dyn-118-139-40-0 ~ % head -n 10 step3_filtered.csv
D       ,sold_time,sold_type        ,sold_price     ,address
                                                                        ,beds ,area
      ,property_type                 ,description
94290,27/03/2021,auction          ,1655000,1/53 Nimmo Street Essendon VIC 3040
                                                              ,5,         ,Townhouse
                     ,Property Description Family Flexibility With A Luxury Edge
n immaculate home of distinction and quality with bright open spaces at every t
rn this extensive entertainers residence is a showpiece of contemporary eleganc
 and premium family living. Designed with flexibility and generosity in mind it
features open-plan living and dining an elaborate kitchen and butlers pantry ea
h with Smeg appliances up to five bedrooms or four plus home office three sleek
fully-tiled bathrooms a first floor retreat and the luxury of two undercover al
resco options with heating.     Read less
36833,15/07/2021,private treaty   ,846000,1/1A Mitchell Street Maribyrnong VIC
032                                                          ,4,         ,Townhouse
                    ,Property Description Luxury Town Home With A Roof Top Terrac
```

iii. Area

CODE

```
awk -F',' 'BEGIN {OFS=","} NR==1 {print; next} $7 > 300 {print}'
step3_filtered.csv > final_filtered.csv
```

EXPLANATION

Here we set the condition that column 7 Area is greater than 300
Now to see the first and last sold times we use below code-

```
% echo "First sold time:"
head -n 2 final_filtered.csv | tail -n 1 | cut -d',' -f2
echo "Last sold time:"
tail -n 1 final_filtered.csv | cut -d',' -f2
```

ANSWER

```
First sold time:
4/11/2021
Last sold time:
15/11/2021
```

TASK- B

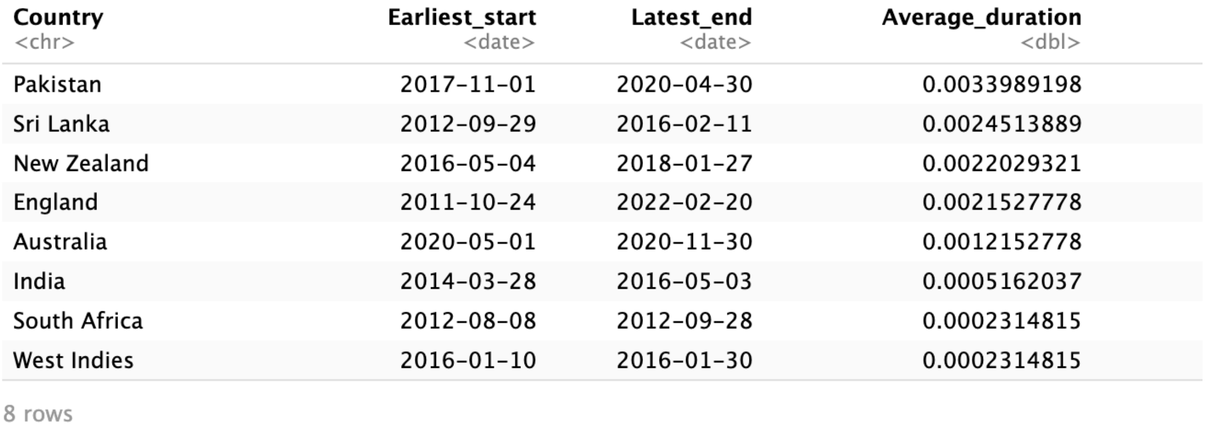
This section focuses on web scrapping. I will be addressing the same by first presenting the code then output. I will proceed to interpret the output and finally explain the code.

Task-B1

CODE:

```
# Load required libraries
library(rvest)
library(dplyr)
library(lubridate)

# URL of the Wikipedia page
url <- "https://en.wikipedia.org/wiki/ICC_Men%27s_T20I_Team_Rankings"

# Read the HTML content from the URL
page <- read_html(url)

# Extract all tables on the page with the class "wikitable"
tables <- page %>% html_nodes("table.wikitable")


# Identify the correct table index (based on inspection, it's the 6th table)
table_index <- 6
historical_rankings_table <- tables[[table_index]] %>% html_table(fill = TRUE)

# Clean up column names
colnames(historical_rankings_table) <- c("Country", "Start", "End", "Duration",
"Cumulative", "Highest_Rating")

# Custom function to parse dates
parse_date <- function(date_str) {
  formats <- c("%d %B %Y", "%d %b %Y", "%Y-%m-%d")
  for (fmt in formats) {
    parsed_date <- as.Date(date_str, format = fmt)
    if (!is.na(parsed_date)) {
      return(parsed_date)
    }
  }
  return(NA)
}

# Convert the Start and End columns to Date format
historical_rankings_table$Start <- sapply(historical_rankings_table$Start,
parse_date)
historical_rankings_table$End <- sapply(historical_rankings_table$End,
parse_date)

# Check for NA values and filter them out
historical_rankings_table <- historical_rankings_table %>%
  filter(!is.na(Start) & !is.na(End))

# Calculate the duration
historical_rankings_table$Duration <-
as.numeric(difftime(historical_rankings_table$End,
historical_rankings_table$Start, units = "days"))

# Summarize the data
summary_table <- historical_rankings_table %>%
  group_by(Country) %>%
  summarize(
    Earliest_start = as.Date(min(Start, na.rm = TRUE)),
    Latest_end = as.Date(max(End, na.rm = TRUE)),
    Average_duration = mean(Duration, na.rm = TRUE)
  ) %>%
  arrange(desc(Average_duration))

# Print the summarized table
print(summary_table)
```

OUTPUT:



```
Country                Earliest_start   Latest_end   Average_duration
<chr>                          <date>       <date>              <dbl>
Pakistan                   2017-11-01   2020-04-30       0.0033989198
Sri Lanka                  2012-09-29   2016-02-11       0.0024513889
New Zealand                2016-05-04   2018-01-27       0.0022029321
England                    2011-10-24   2022-02-20       0.0021527778
Australia                  2020-05-01   2020-11-30       0.0012152778
India                      2014-03-28   2016-05-03       0.0005162037
South Africa               2012-08-08   2012-09-28       0.0002314815
West Indies                2016-01-10   2016-01-30       0.0002314815
8 rows
```

We can see the above table has been returned from web scrapping retaining all the correct date formats as specified.

EXPLANATION:

In the first part of the code we will load the necessary libraries and their uses, which are:

- rvest: Web scraping
- dplyr: Data Manipulation
- lubridate: Dates

The next step is to read the html url by dubbing the name as url and then using `read_html` command.

After this, we extract the tables using html_nodes and identify the correct table that we want to work with which is the 6th one for this task.Once that is done we convert the table into a dataframe using `tables[[table_index]]%>%html_table(fill=TRUE):`
Then we clean up the column names for mor clarity and parse the dates to change them into a custom format.We will also filter out any NA values using `filter(!is.na(Start) & ! is.na(End))`
After this we calculate the duration by taking the difference between the start and end dates and finally summarise and print the table.

Task-B2

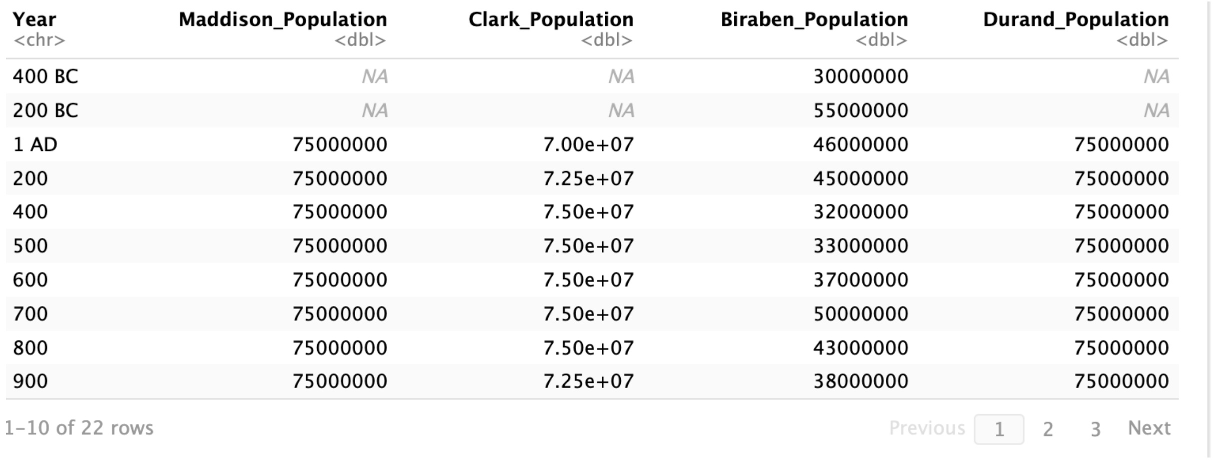
For the purpose of this assignment I have chosen to explore the trends in the Indian Demographic Population using the link:
https://en.wikipedia.org/wiki/Demographics_of_India

CODE:

```
# Load required libraries
library(rvest)
library(dplyr)
library(ggplot2)
library(scales)
library(tidyr)
```

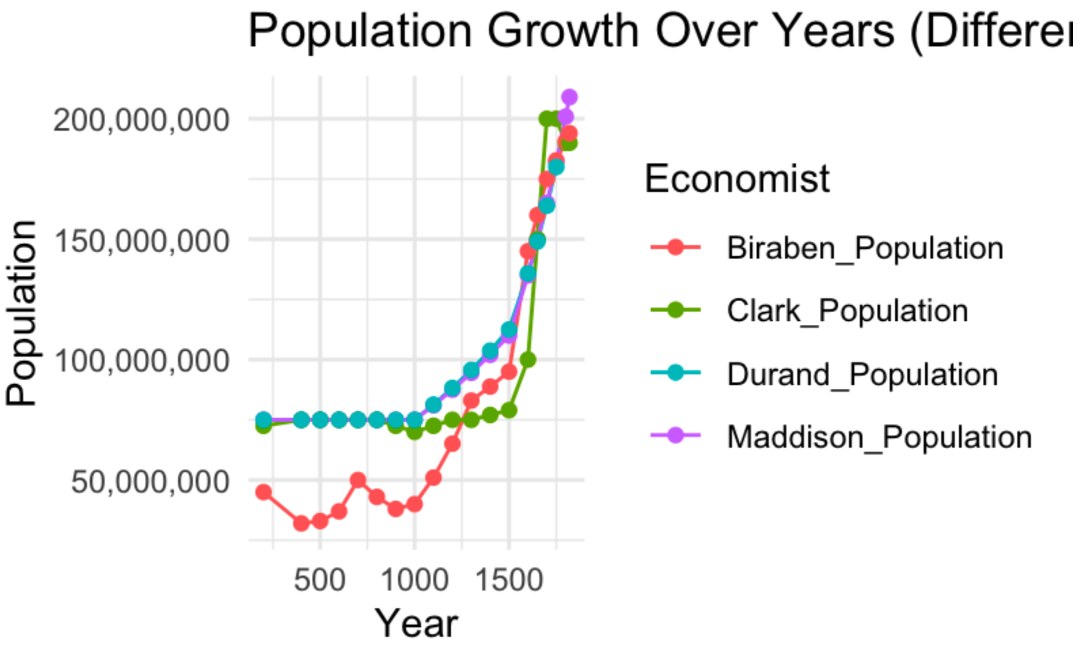
```
# URL of the Wikipedia page
url <- "https://en.wikipedia.org/wiki/Demographics_of_India"

# Read the HTML content from the URL
page <- read_html(url)

# Extract all tables on the page with the class "wikitable"
tables <- page %>% html_nodes("table.wikitable")

# Identify the correct table index (based on inspection, it's the 1st
table)
table_index <- 1
demographics_table <- tables[[table_index]] %>% html_table(fill = TRUE)

# Clean up column names
colnames(demographics_table) <- c("Year", "Maddison_Population",
"Maddison_Growth", "Clark_Population", "Clark_Growth",
"Biraben_Population", "Biraben_Growth", "Durand_Population",
"Durand_Growth", "McEvedy_Population", "McEvedy_Growth")

# Select only the columns Year, Maddison Population, Clark Population,
Biraben Population, and Durand Population
population_table <- demographics_table %>% select(Year,
Maddison_Population, Clark_Population, Biraben_Population,
Durand_Population)

# Convert populations to numeric, removing any commas or non-numeric
characters
population_table <- population_table %>%
  mutate(across(c(Maddison_Population, Clark_Population,
Biraben_Population, Durand_Population), ~ as.numeric(gsub(",", "",
gsub("[^0-9]", "", .)))))

# Filter out rows with NA values in all population columns
population_table <- population_table %>% filter(!is.na(Maddison_Population)
| !is.na(Clark_Population) | !is.na(Biraben_Population) |
!is.na(Durand_Population))

# Print the table containing the selected population columns
print(population_table)

# Transform the data to a long format for plotting
population_long <- population_table %>%
  pivot_longer(cols = c(Maddison_Population, Clark_Population,
Biraben_Population, Durand_Population), names_to = "Economist", values_to
= "Population")

# Create a line plot using ggplot2
ggplot(population_long, aes(x = as.numeric(Year), y = Population, color =
Economist)) +
  geom_line() +
  geom_point() +
  scale_y_continuous(labels = comma) + # Format y-axis labels to avoid
exponents
  labs(title = "Population Growth Over Years (Different Economists)", x =
"Year", y = "Population") +
  theme_minimal()
```

OUTPUT:



Year <chr>	Maddison_Population <dbl>	Clark_Population <dbl>	Biraben_Population <dbl>	Durand_Population <dbl>
400 BC	NA	NA	30000000	NA
200 BC	NA	NA	55000000	NA
1 AD	75000000	7.00e+07	46000000	75000000
200	75000000	7.25e+07	45000000	75000000
400	75000000	7.50e+07	32000000	75000000
500	75000000	7.50e+07	33000000	75000000
600	75000000	7.50e+07	37000000	75000000
700	75000000	7.50e+07	50000000	75000000
800	75000000	7.50e+07	43000000	75000000
900	75000000	7.25e+07	38000000	75000000

1–10 of 22 rows



INTERPRETATION

The graph displays the population growth over the years as estimated by different economists: Maddison, Clark, Biraben, and Durand. The x-axis represents the years, while the y-axis shows the population. Each line and corresponding points denote the population estimates provided by a specific economist. The data reveals a consistent upward trend in population across all economists' estimates, with significant increases starting around the 1500s, reflecting historical population growth trends. The variations between economists' estimates highlight the differences in historical population data interpretation and methodology.

EXPLANATION

Here, the additional libraries loaded:

- **ggplot2**: Used to visualize data.
- **scales**: Used for properly scaling data into correct formats for scales.

- **tidyr**: Data transformation.

Similar to previous task, first we will read the url and extract the data from the correct table (Here it is table 1).
After that, we clean up the column names after which we select and clean the population data by selecting only relevant columns, in this case we wanted only first 4 economists. Ensure there are no NA values and then finally print out the table.
Once the table has been printed, we transform it to use it to make plots since long formats work better with ggplot2.
Lastly, we use ggplot to plot out a comparative line graph.

TASK-C

Exploratory Analysis of tweets using R

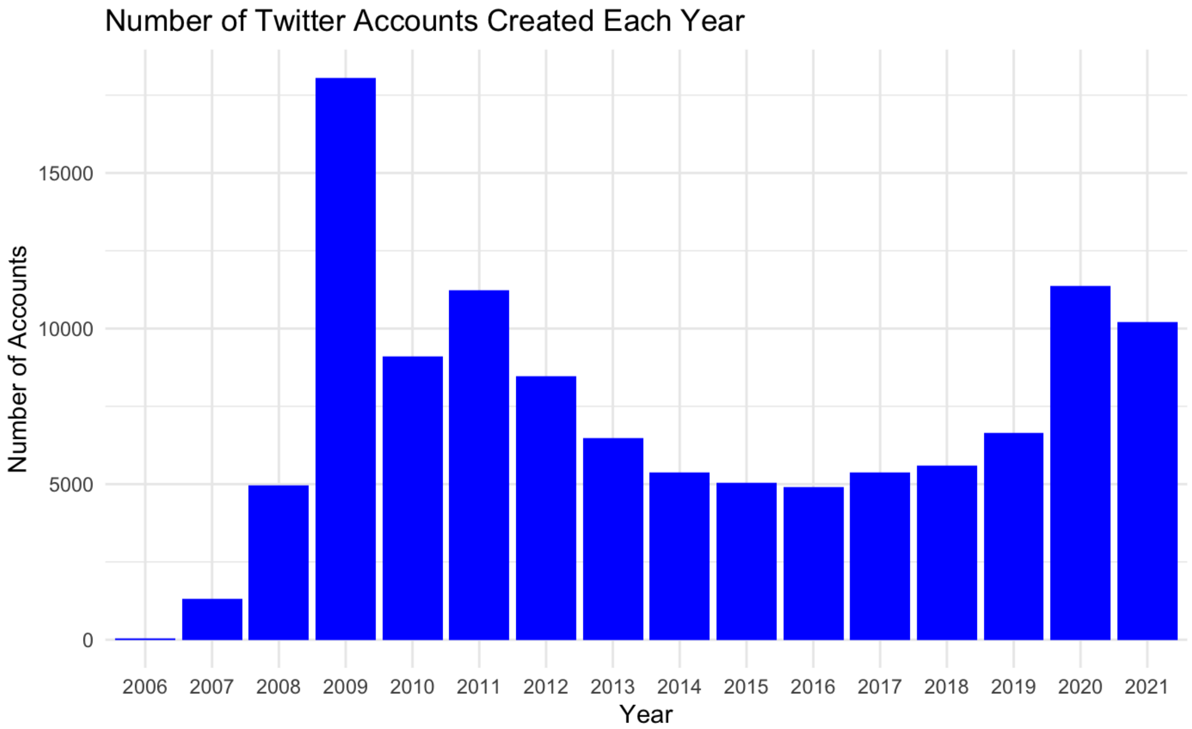
1. Tweet Creators

   1.1. Accounts over the Years

CODE:

```
# Load required libraries
library(ggplot2)
library(dplyr)
library(stringr)
library(tidyverse)
library(tidytext)
library(textclean)

# Read the CSV file
tweets <- read.csv("Olympics_tweets.csv", stringsAsFactors = FALSE)

# Extract the year from the user_created_at column
tweets$year <- format(as.Date(tweets$user_created_at, format =
"%d/%m/%Y"), "%Y")

# Remove NA values in the year column
tweets <- tweets %>% filter(!is.na(year))

# Count the number of accounts created each year
yearly_counts <- tweets %>%
  group_by(year) %>%
  summarise(count = n())

# Create the bar chart
ggplot(yearly_counts, aes(x = year, y = count)) +
  geom_bar(stat = "identity", fill = "blue") +
  labs(title = "Number of Twitter Accounts Created Each Year",
       x = "Year",
       y = "Number of Accounts") +
  theme_minimal()
```

OUTPUT:



From the above chart, we observe that number of accounts peak around 2009 which indicates a surge of new accounts being created then after which the number fluctuates over the years with significant increases again around 2011 and 2020.It declines sometimes as seen in 2013 and 2019. The graph shows us the account trends over the years.

EXPLANATION

First we begin by loading libraries:

- ggplot2: Used to visualize data.
- dplyr: Data Manipulation
- stringr: String Manipulation
- tidyverse:Library for datascience
- tidytext:Text mining
- textclean: Text Cleaning

First, we read from the CSV File after which we format and extract the year part of the date and remove NA Values.
We then count the number of accounts each year using command-
yearly_counts <- tweets %>% group_by(year) %>% summarise(count = n())
Lastly, we use ggplot to build visualisations.

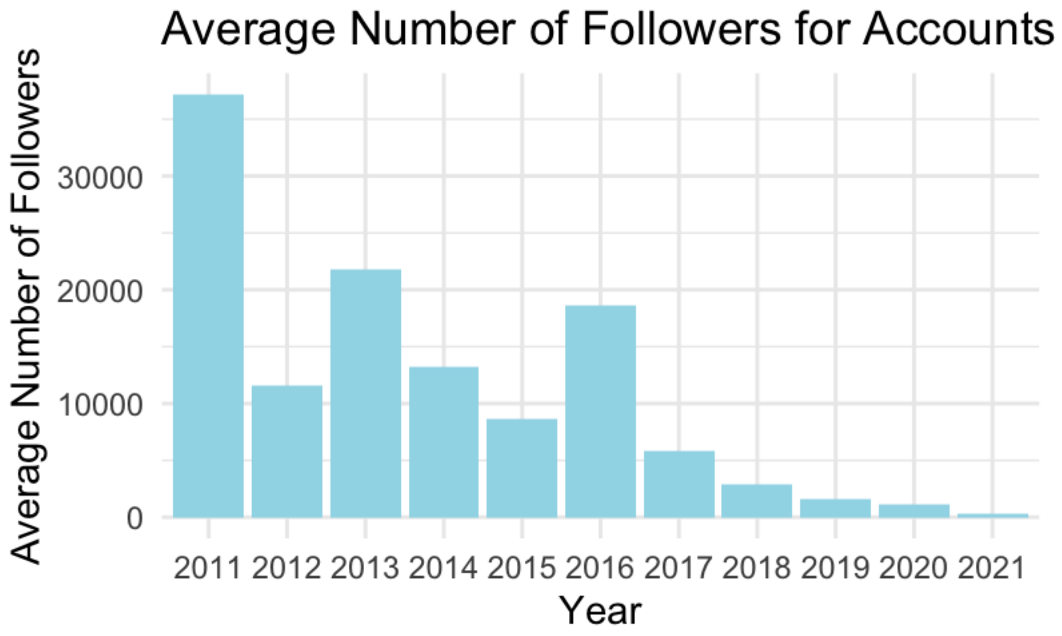
1.2.User_followers

CODE

```
# Extract the year from the user_created_at column
tweets$year <- format(as.Date(tweets$user_created_at, format = "%d/%m/%Y"), "%Y")

# Convert year to numeric and filter for accounts created after 2010
tweets <- tweets %>% filter(as.numeric(year) > 2010)
```

```
# Remove NA values in the year and user_followers columns
tweets <- tweets %>% filter(!is.na(year) & !is.na(user_followers))

# Calculate the average number of user_followers for each year
yearly_avg_followers <- tweets %>%
  group_by(year) %>%
  summarise(avg_followers = mean(user_followers, na.rm = TRUE))

# Create the bar chart
ggplot(yearly_avg_followers, aes(x = year, y = avg_followers)) +
  geom_bar(stat = "identity", fill = "lightblue") +
  labs(title = "Average Number of Followers for Accounts Created After 2010",
       x = "Year",
       y = "Average Number of Followers") +
  theme_minimal()
```

OUTPUT:



We can see a fluctuation of average number of followers till 2016 after which we see a steady decline there on.

EXPLANATION

I have omitted from this chunk the code that was already in the previous chunk like libarary and data loading to reduce redundancy. Here we extract the year and follow similar steps as before by removing NA values after which we calculate average user followers for each year and use ggplot to generate a bar chart.To be more in detail:

- □ tweets <- tweets %>% filter(as.numeric(year) > 2010) is the line that is used to filter accounts created after 2016 after converting year to numeric.
- □ tweets <- tweets %>% filter(!is.na(year) & !is.na(user_followers)) removes NA values.
- □ yearly_avg_followers <- tweets %>%
  group_by(year) %>%
  summarise(avg_followers = mean(user_followers, na.rm = TRUE)): This bit of code essentially calculates the average number of followers for each year.
- □ Lastly, we use ggplot to visualise a bachart by providing it details and aesthetics like filling the bars with lightblue colour.

1.3. Graphs

Number of Twitter Accounts Created Each Year:

- There was a significant spike in the number of Twitter accounts created in 2009.
- From 2009 to 2012, the number of accounts created each year remained relatively high.
- There is a noticeable dip in the number of accounts created from 2013 to 2019, followed by a slight increase in 2020 and 2021.

Average Number of Followers for Accounts Created After 2010:

- Accounts created in 2011 have the highest average number of followers.
- There is a general decline in the average number of followers for accounts created from 2012 onwards.
- The average number of followers for accounts created in recent years (2017 onwards) is significantly lower compared to earlier years.

Potential Explanations:

- Spike in 2009 Account Creations as Twitter gained immense popularity around 2009, possibly due to increased media coverage and adoption by celebrities and public figures, leading to a surge in new accounts.

- Decline in Average Followers Over Time:
  - Early adopters of Twitter (accounts created around 2011) were likely to gain more followers as the platform was less saturated, and they had more time to build their follower base.
  - As Twitter became more mainstream and the number of users grew, the competition for followers increased, making it harder for newer accounts to amass large follower counts.
  - Changes in Twitter's algorithm and user behaviour over the years could also contribute to the declining average number of followers for newer accounts.

- Slight Increase in Recent Years:
  - The slight increase in the number of accounts created in 2020 and 2021 could be attributed to the global events such as the COVID-19 pandemic, where people sought more online interactions and social media usage surged.

Overall, these observations suggest that the dynamics of Twitter's user growth and engagement have evolved significantly over the years, influenced by various external factors and the platform's own development.

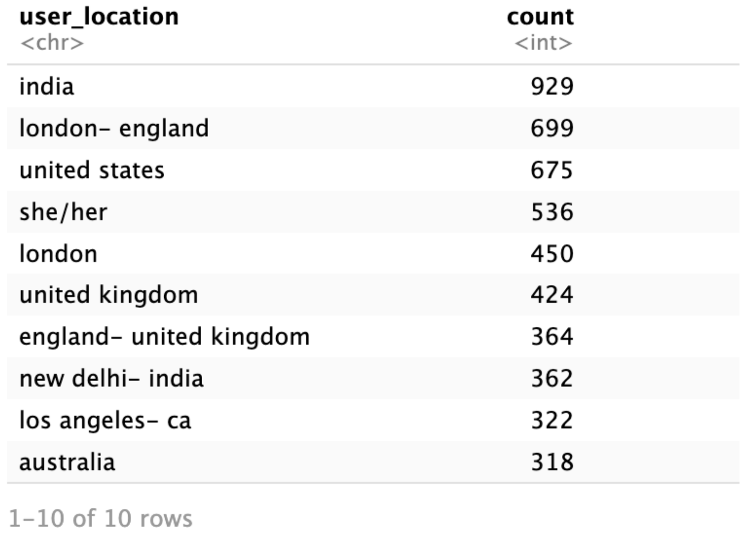
1.4. Locations

CODE:

```
# Clean the user_location column (trim whitespace, convert to lowercase)
tweets$user_location <- trimws(tolower(tweets$user_location))

# Count the occurrences of each location and display the top 10 most frequent
locations
location_counts <- tweets %>%
  filter(!is.na(user_location) & user_location != "") %>%
  group_by(user_location) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(10)

# Print the top 10 most frequent locations
print(location_counts)
```

```
# Count the total number of tweets associated with these top 10 most frequent
locations
total_tweets_top10_locations <- sum(location_counts$count)
cat("Total number of tweets associated with the top 10 most frequent locations:",
total_tweets_top10_locations, "\n")

#visualise
# Clean the user_location column (trim whitespace, convert to lowercase)
tweets$user_location <- trimws(tolower(tweets$user_location))

# Count the occurrences of each location and display the top 10 most frequent
locations
location_counts <- tweets %>%
  filter(!is.na(user_location) & user_location != "") %>%
  group_by(user_location) %>%
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(10)

# Calculate percentages
location_counts <- location_counts %>%
  mutate(percentage = count / sum(count) * 100)

# Print the top 10 most frequent locations with percentages
print(location_counts)

# Create a pie chart with percentages
ggplot(location_counts, aes(x = "", y = percentage, fill = user_location)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  geom_text(aes(label = sprintf("%.1f%%", percentage)),
            position = position_stack(vjust = 0.5)) +
  labs(title = "Top 10 Most Frequent User Locations",
       fill = "User Location") +
  theme_void()
```

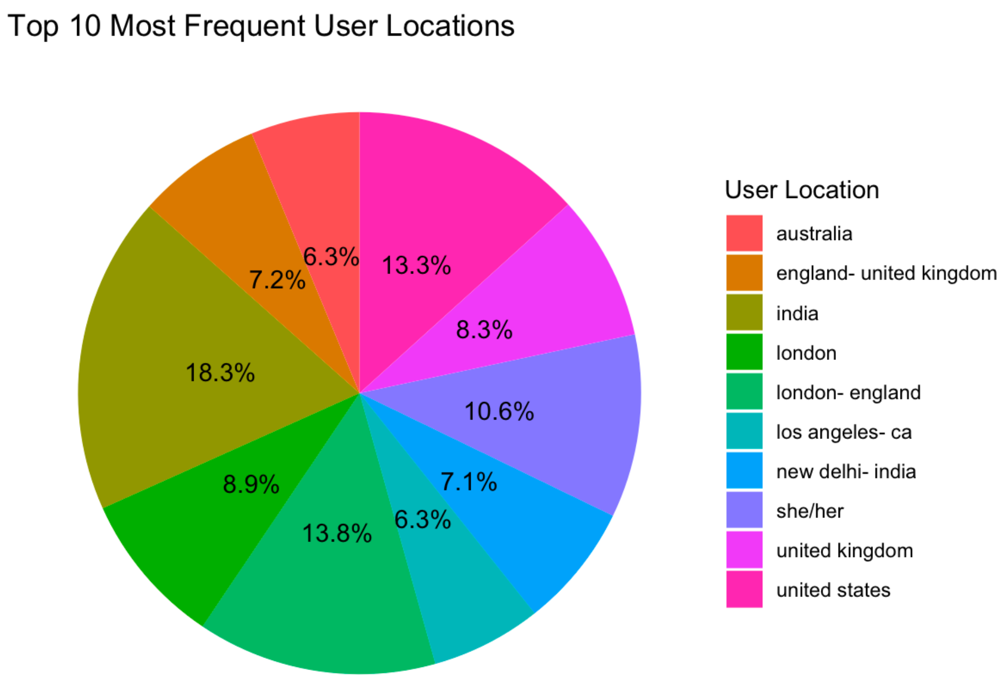
OUTPUT:



user_location <chr>	count <int>
india	929
london- england	699
united states	675
she/her	536
london	450
united kingdom	424
england- united kingdom	364
new delhi- india	362
los angeles- ca	322
australia	318

1-10 of 10 rows

```
Total number of tweets associated with the top 10 most frequent locations: 5079
```



As we can see from above outputs, it appears India holds #1 position as most frequent user location and USA holds the 10th position. There is a total of 5079 tweets are associated with these top ten locations.

EXPLANATION:

First, we begin by cleaning the user_location column to remove any white space present by using the timws function and standardising everything in lowercase.
After that we proceed to count the occurences of each location and display top 10 most frequent locations-

- Filtering: filter(!is.na(user_location) & user_location != "") removes rows where user_location is NA or empty.
- Grouping: group_by(user_location) groups the data by user_location.
- Summarizing: summarise(count = n()) counts the occurrences of each location.
- Sorting: arrange(desc(count)) sorts the counts in descending order.
- Selecting Top 10: head(10) selects the top 10 most frequent locations.

Lastly, we calculate percentages and use ggplot to give us a pie chart.

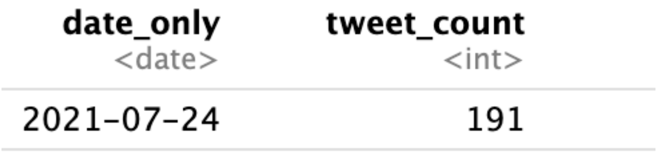
2. Tweet Analysis

2.1. Least Tweets

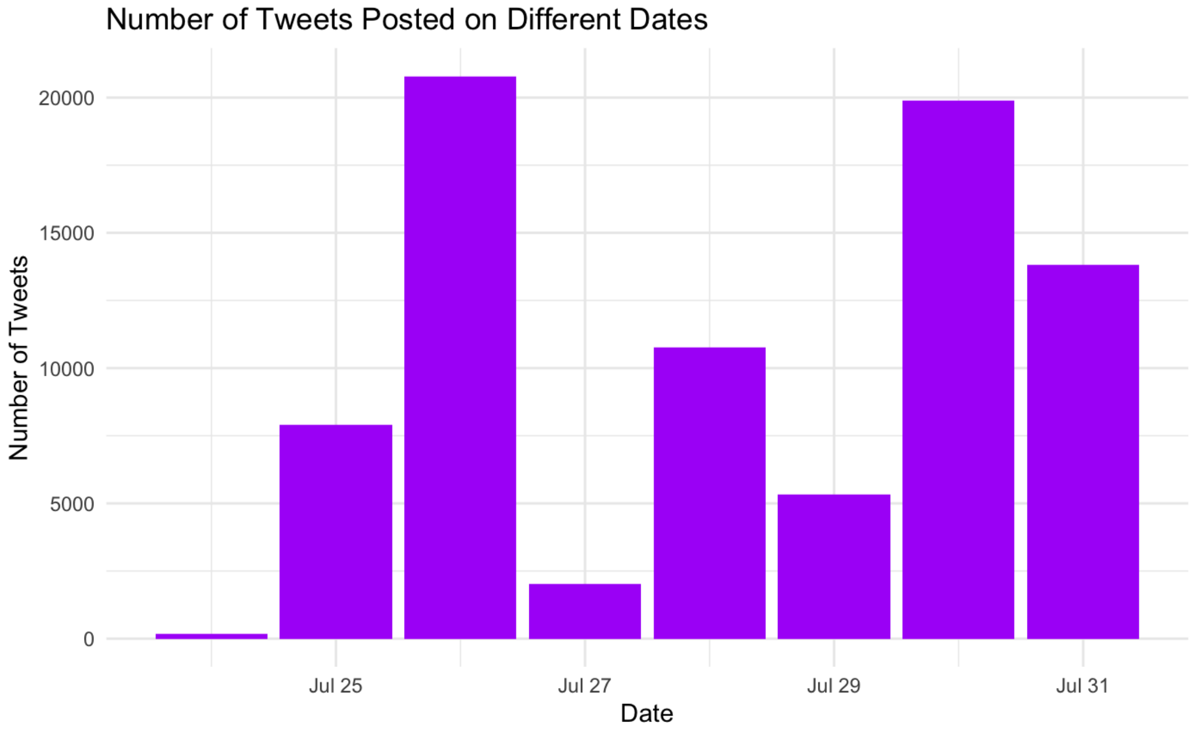
CODE:

```
# Extract the date from the 'date' column
tweets$date_only <- as.Date(tweets$date, format = "%d/%m/%Y %H:%M")

# Count the number of tweets for each date
date_counts <- tweets %>%
  group_by(date_only) %>%
  summarise(tweet_count = n()) %>%
  arrange(tweet_count)

# Print the date with the lowest number of tweets
print(date_counts[1, ])

# Create a bar chart
ggplot(date_counts, aes(x = date_only, y = tweet_count)) +
  geom_bar(stat = "identity", fill = "purple") +
  labs(title = "Number of Tweets Posted on Different Dates",
       x = "Date",
       y = "Number of Tweets") +
  theme_minimal()
```

OUTPUT:



date_only <date>	tweet_count <int>
2021-07-24	191



From above graph, we can conclude that the date with least tweets is Jun 24th.

EXPLANATION

This code begins by extracting the date part of the date column and then proceeds to count the number of tweets for each date as seen below:

- □ Grouping: group_by(date_only) groups the data by the extracted date_only column.
- □ Summarizing: summarise(tweet_count = n()) counts the number of tweets for each date.

- Sorting: arrange(tweet_count) sorts the dates by the number of tweets in ascending order.

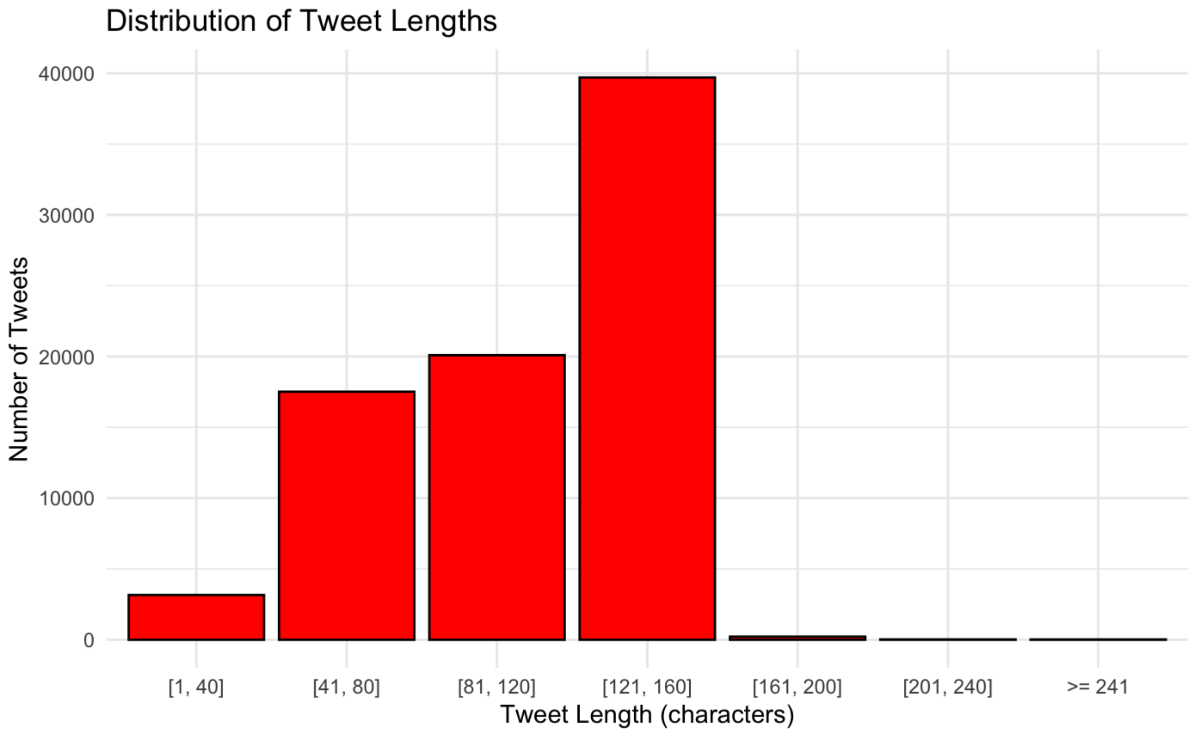
Then print the date with least number of tweets using `print(date_counts[1, ])`
Lastly, we use ggplot and fill the bars with purple colour to visualise the same.

2.2. Text Length

CODE

```
# Calculate the length of the text in each tweet
tweets$text_length <- nchar(tweets$text)

# Categorize the tweet lengths
tweets$tweet_length_category <- cut(tweets$text_length,
                                    breaks = c(0, 40, 80, 120, 160,
200, 240, Inf),
                                    labels = c("[1, 40]", "[41,
80]", "[81, 120]", "[121, 160]", "[161, 200]", "[201, 240]", ">=
241"),
                                    right = FALSE)

# Create a bar chart with the categorized tweet lengths
ggplot(tweets, aes(x = tweet_length_category)) +
  geom_bar(fill = "red", color = "black") +
  labs(title = "Distribution of Tweet Lengths",
       x = "Tweet Length (characters)",
       y = "Number of Tweets") +
  theme_minimal() +
  scale_x_discrete(drop = FALSE)
```

OUTPUT:



From above graph, we can conclude that tweets with character length 121-160 have the highest number of tweets and least are more than 241 characters.

EXPLANATION:

In this code we start by calculating the length of text in each tweet then we proceed to categorize them as seen below:

- Cut Function: The cut function categorizes the tweet lengths into bins.
- Breaks: The breaks parameter specifies the boundaries of the bins: 0-40, 41-80, 81-120, 121-160, 161-200, 201-240, and 241 or more characters.
- Labels: The labels parameter assigns descriptive labels to each bin.
- Right Parameter: Setting right = FALSE ensures that the intervals are left-closed (inclusive of the left endpoint but not the right).

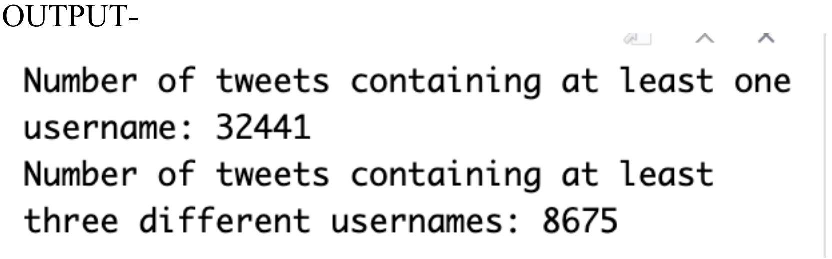
Lastly, we use ggplot to help output a bar graph.

2.3.Twitter mentions

CODE-

```
# Define a function to count unique usernames in a tweet
count_usernames <- function(text) {
  usernames <- str_extract_all(text, "@\\w+")[[1]]
  unique_usernames <- unique(usernames)
  return(length(unique_usernames))
}

# Apply the function to each tweet to count the usernames
tweets$username_count <- sapply(tweets$text, count_usernames)

# Count the number of tweets containing at least one username
tweets_with_usernames <- sum(tweets$username_count >= 1)

# Count the number of tweets containing at least three different usernames
tweets_with_three_usernames <- sum(tweets$username_count >= 3)

# Print the results
cat("Number of tweets containing at least one username:",
tweets_with_usernames, "\n")
cat("Number of tweets containing at least three different usernames:",
tweets_with_three_usernames, "\n")
```

OUTPUT-



```
Number of tweets containing at least one
username: 32441
Number of tweets containing at least
three different usernames: 8675
```

EXPLANATION

First, we define a function to count unique usernames in a tweet:

- Extract Usernames: str_extract_all(text, "@\\w+")[[1]] uses a regular expression to find all occurrences of usernames (strings starting with '@' followed by word characters) in the text. The [[1]] extracts the list of matches from the resulting list.

- Unique Usernames: unique(usernames) identifies unique usernames from the extracted list.
- Count Usernames: return(length(unique_usernames)) returns the count of unique usernames in the tweet.

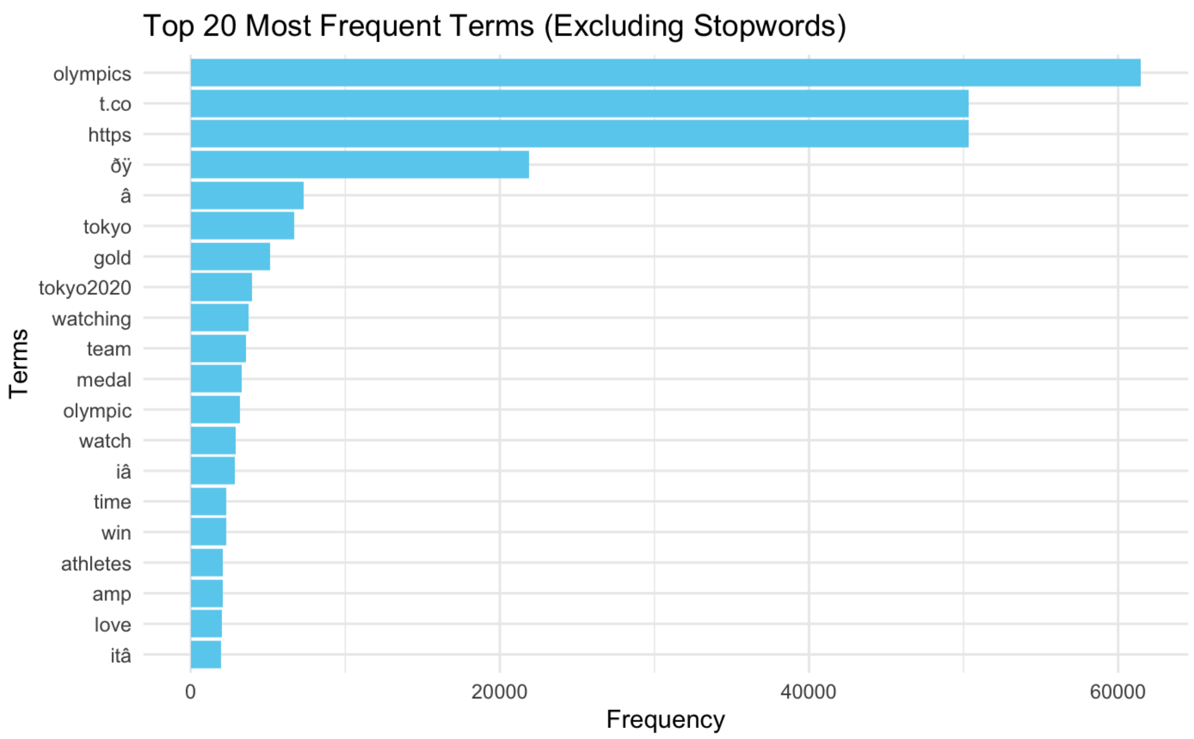
Then we apply this function to each tweet and proceed to count the number of tweets containing usernames which we filter out to find those containing atleast 3 and print out the results.

2.4.Frequent words

CODE-

```
# Create a tibble and tokenize the text data
tweets_tidy <- tweets %>%
  select(text) %>%
  unnest_tokens(word, text)

# Calculate term frequency including stopwords
term_frequency <- tweets_tidy %>%
  count(word, sort = TRUE)

# Get the top 20 most frequent terms including stopwords
top_20_terms <- term_frequency %>% head(20)

# Remove stopwords
data("stop_words")
tweets_tidy_no_stopwords <- tweets_tidy %>%
  anti_join(stop_words, by = "word")

# Calculate term frequency excluding stopwords
term_frequency_no_stopwords <- tweets_tidy_no_stopwords %>%
  count(word, sort = TRUE)

# Get the top 20 most frequent terms excluding stopwords
top_20_terms_no_stopwords <- term_frequency_no_stopwords %>% head(20)

# Print the results
cat("Top 20 most frequent terms (including stopwords):\n")
print(top_20_terms)

cat("\nTop 20 most frequent terms (excluding stopwords):\n")
print(top_20_terms_no_stopwords)

# Visualize the top 20 terms excluding stopwords
top_20_terms_no_stopwords %>%
  ggplot(aes(x = reorder(word, n), y = n)) +
  geom_col(fill = "skyblue") +
  coord_flip() +
  labs(title = "Top 20 Most Frequent Terms (Excluding Stopwords)",
       x = "Terms",
       y = "Frequency") +
  theme_minimal()
```

OUTPUT:



The bar chart displays the top 20 most frequent terms in the tweets about the Olympics, excluding common stopwords. The terms "olympics," "t.co," and "https" are the most frequent, indicating frequent mentions of the event and many tweets containing links. Other notable terms include "tokyo," "gold," "team," "medal," and "athletes," reflecting discussions around the Tokyo 2020 Olympics, athletes, and their achievements.

EXPLANATION
tweets_tidy <- tweets %>% select(text) %>% unnest_tokens(word, text): This line creates a tidy tibble y selecting the text column from the tweets dataframe and tokenizing the text into individual words.Then we split each text into different words.after which we calculate the frequency of terms along with stop words. After this we get the top 20 most frequent terms and remove the stopwords by using `tweets_tidy_no_stopwords <- tweets_tidy %>% anti_join(stop_words, by = "word")`
Then we calculate most frequent terms and output the same using ggplot

TASK-D

1. Features
   Proposed Features

   Here are some potential features that could be engineered to help predict the usefulness of a dialogue:

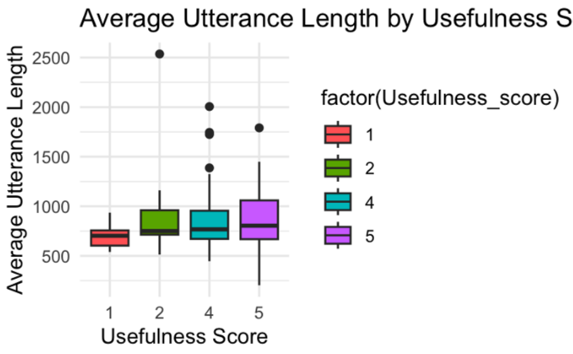
   1. Average Utterance Length: The average length of utterances in a dialogue.
   2. Number of Utterances: The total number of utterances in a dialogue.
   3. Number of Chatbot Utterances: The number of utterances made by the chatbot.
   4. Number of Student Utterances: The number of utterances made by the student.
   5. Response Time: The average time between student and chatbot responses.

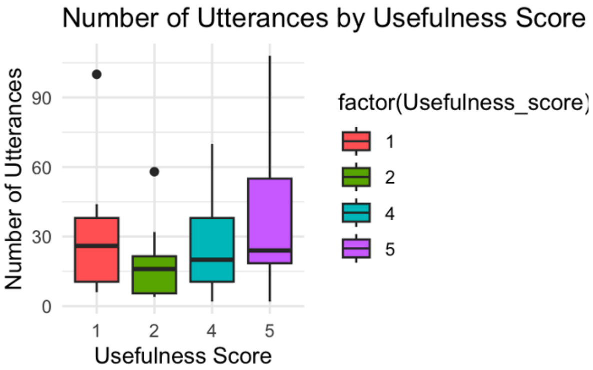
For this task, we'll select Average Utterance Length and Number of Utterances to visualize.

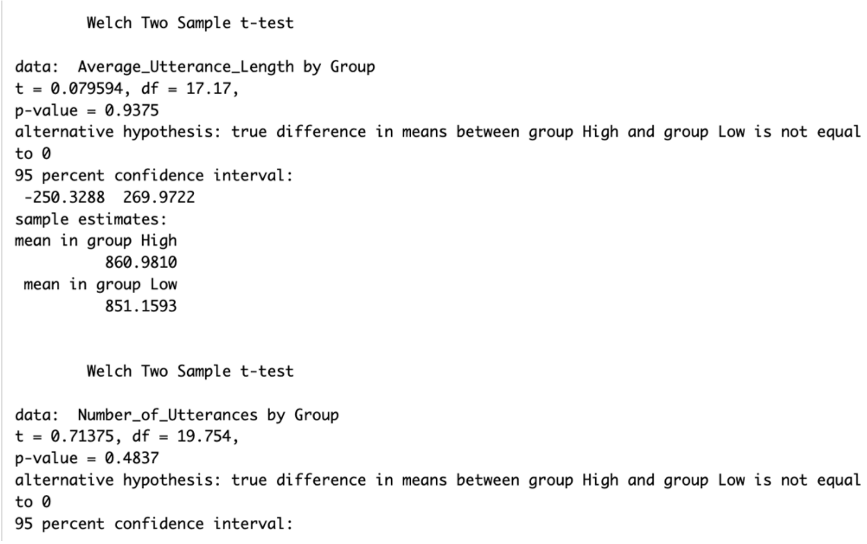
CODE:

```
# Load required libraries
library(readr)
library(dplyr)
library(ggplot2)

# Load the data files
utterance_train <- read_csv('dialogue_utterance_train.csv')
usefulness_train <- read_csv('dialogue_usefulness_train.csv')

# Correct the column names
colnames(utterance_train) <- c("Dialogue_ID", "Timestamp", "Interlocutor", "Utterance_text")
colnames(usefulness_train) <- c("Dialogue_ID", "Usefulness_score")

# Merge the data on Dialogue_ID
merged_data <- utterance_train %>%
  inner_join(usefulness_train, by = "Dialogue_ID")

# Feature Engineering: Calculate Average Utterance Length and Number of Utterances
features <- merged_data %>%
  group_by(Dialogue_ID) %>%
  summarize(
    Average_Utterance_Length = mean(nchar(Utterance_text)),
    Number_of_Utterances = n(),
    Usefulness_score = first(Usefulness_score)
  )

# Filter the data for visualization
filtered_data <- features %>%
  filter(Usefulness_score %in% c(1, 2, 4, 5))

# Create boxplots
p1 <- ggplot(filtered_data, aes(x = factor(Usefulness_score), y = Average_Utterance_Length, fill =
factor(Usefulness_score))) +
  geom_boxplot() +
  labs(title = "Average Utterance Length by Usefulness Score", x = "Usefulness Score", y = "Average
Utterance Length") +
  theme_minimal()

p2 <- ggplot(filtered_data, aes(x = factor(Usefulness_score), y = Number_of_Utterances, fill =
factor(Usefulness_score))) +
  geom_boxplot() +
  labs(title = "Number of Utterances by Usefulness Score", x = "Usefulness Score", y = "Number of
Utterances") +
  theme_minimal()

# Print the plots
print(p1)
print(p2)

# Group the data into two categories for t-test
grouped_data <- filtered_data %>%
  mutate(Group = case_when(
    Usefulness_score %in% c(1, 2) ~ "Low",
    Usefulness_score %in% c(4, 5) ~ "High"
  ))

# Perform t-test for Average Utterance Length
t_test_length <- t.test(Average_Utterance_Length ~ Group, data = grouped_data)
print(t_test_length)

# Perform t-test for Number of Utterances
t_test_utterances <- t.test(Number_of_Utterances ~ Group, data = grouped_data)
print(t_test_utterances)
```

OUTPUT







```
        Welch Two Sample t-test

data:  Average_Utterance_Length by Group
t = 0.079594, df = 17.17,
p-value = 0.9375
alternative hypothesis: true difference in means between group High and group Low is not equal
to 0
95 percent confidence interval:
 -250.3288  269.9722
sample estimates:
mean in group High
          860.9810
 mean in group Low
          851.1593


        Welch Two Sample t-test

data:  Number_of_Utterances by Group
t = 0.71375, df = 19.754,
p-value = 0.4837
alternative hypothesis: true difference in means between group High and group Low is not equal
to 0
95 percent confidence interval:
```

<u>Average Utterance Length by Usefulness Score:</u>
The graph shows that dialogues with a usefulness score of 5 tend to have longer average utterance lengths compared to other scores. There is a noticeable spread in the data for scores 4 and 5, indicating variability in utterance lengths for these higher usefulness scores.

<u>Number of Utterances by Usefulness Score:</u>
The graph indicates that dialogues with a usefulness score of 5 also tend to have a higher number of utterances. There is a trend of increasing number of utterances with increasing usefulness score, with the highest scores showing the most variability.

The Welch Two Sample t-tests indicate no significant difference in means for both average utterance length (p=0.9375) and number of utterances (p=0.4837) between high and low usefulness groups.

EXPLANATION

This code creates a Random Forest model to predict the usefulness score of dialogues based on features derived from dialogue utterances. We start by loading necessary libraries required for data manipulation, reprocessing and generation of random forest. After that we load the data and standardise the column names after which we create a function to calculate features named calculate_features. Then we apply the feature to the test data and rescale the features to predict the usefulness scores and merge them with the test data and verifying changes.

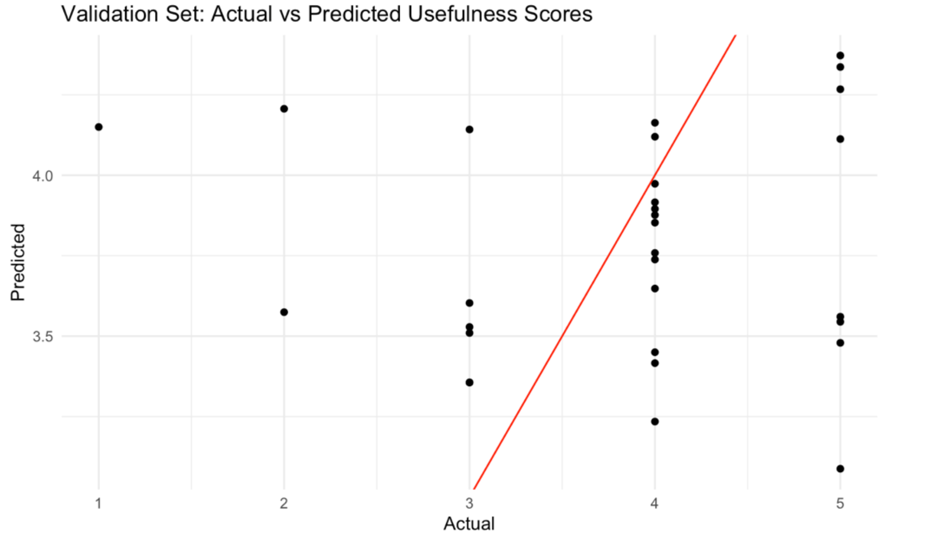
2. Model

CODE

```
# Load required libraries
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(caret)
library(randomForest)

# Load the data files
utterance_train <- read_csv('dialogue_utterance_train.csv')
usefulness_train <- read_csv('dialogue_usefulness_train.csv')
utterance_validation <- read_csv('dialogue_utterance_validation.csv')
usefulness_validation <- read_csv('dialogue_usefulness_validation.csv')
utterance_test <- read_csv('dialogue_utterance_test.csv')
usefulness_test <- read_csv('dialogue_usefulness_test.csv')


# Rename columns to standardize names
colnames(utterance_train) <- c("Dialogue_ID", "Timestamp", "Interlocutor",
"Utterance_text")
colnames(utterance_validation) <- c("Dialogue_ID", "Timestamp", "Interlocutor",
"Utterance_text")
colnames(utterance_test) <- c("Dialogue_ID", "Timestamp", "Interlocutor",
"Utterance_text")
colnames(usefulness_train) <- c("Dialogue_ID", "Usefulness_score")
colnames(usefulness_validation) <- c("Dialogue_ID", "Usefulness_score")
colnames(usefulness_test) <- c("Dialogue_ID", "Usefulness_score")

# Function to calculate features
calculate_features <- function(utterance_data, usefulness_data) {
  utterance_data %>%
    group_by(Dialogue_ID) %>%
    summarise(
      Number_of_Utterances = n(),
      Average_Utterance_Length = mean(nchar(Utterance_text), na.rm = TRUE),
      Interlocutor_Ratio = sum(Interlocutor == 'Student') / sum(Interlocutor ==
'Chatbot', na.rm = TRUE)
    ) %>%
    inner_join(usefulness_data, by = "Dialogue_ID")
}

# Calculate features for training and validation sets
train_features <- calculate_features(utterance_train, usefulness_train)
validation_features <- calculate_features(utterance_validation,
usefulness_validation)

# Prepare data for training
train_x <- train_features %>%
  select(Number_of_Utterances, Average_Utterance_Length, Interlocutor_Ratio)
train_y <- train_features$Usefulness_score

# Train Random Forest Model
rf_model <- randomForest(train_x, train_y, ntree = 100, importance = TRUE)

# Print model summary
print(rf_model)

# Prepare validation data
validation_x <- validation_features %>%
  select(Number_of_Utterances, Average_Utterance_Length, Interlocutor_Ratio)
validation_y <- validation_features$Usefulness_score
```

```
# Predict on validation set
validation_pred <- predict(rf_model, validation_x)

# Calculate evaluation metrics
rmse <- sqrt(mean((validation_y - validation_pred)^2))
mae <- mean(abs(validation_y - validation_pred))

# Print evaluation metrics
cat("RMSE: ", rmse, "\n")
cat("MAE: ", mae, "\n")

# Plot predictions vs actual values
ggplot(data = data.frame(Actual = validation_y, Predicted = validation_pred), aes(x
= Actual, y = Predicted)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = 'red') +
  labs(title = "Validation Set: Actual vs Predicted Usefulness Scores", x =
"Actual", y = "Predicted") +
  theme_minimal()
```

OUTPUT



Model Summary:

- Type of random forest: Regression
- Number of trees: 100
- Number of variables tried at each split: 1
- Mean of squared residuals: 1.238932
- % Var explained: -5.56 (This indicates the model is not performing well as a negative percentage indicates that the model performs worse than a simple mean prediction)

Evaluation Metrics:

- RMSE (Root Mean Square Error): 1.065752
- MAE (Mean Absolute Error): 0.7741437

Plot Interpretation:

The scatter plot shows the relationship between the actual usefulness scores and the predicted usefulness scores from the random forest model. The red line represents the line of perfect prediction (where actual equals predicted).

- Points closer to the red line indicate better predictions.
- The spread of points around the line indicates the level of prediction error.
- From the plot, it is evident that the predictions are not very accurate, especially since the points are spread widely and do not align closely with the red line.

Therefore, the random forest model trained with the current features (Number_of_Utterances, Average_Utterance_Length, Interlocutor_Ratio) does not perform well in predicting the usefulness scores. The low % variance explained and the spread of points in the plot indicate that the model has limited predictive power with the given features.

Explanation

Here we start by loading libraries and data as required and standardise the column names and begin to calculate features for each dialogue then do it for training and validation sets. Then we prepare data for training:
train_x <- train_features %>%
   select(Number_of_Utterances, Average_Utterance_Length, Interlocutor_Ratio)
train_y <- train_features$Usefulness_score

After this we train the random forest and prepare data for validation and make predictions for the validation set and plot the predictions vs actual values.

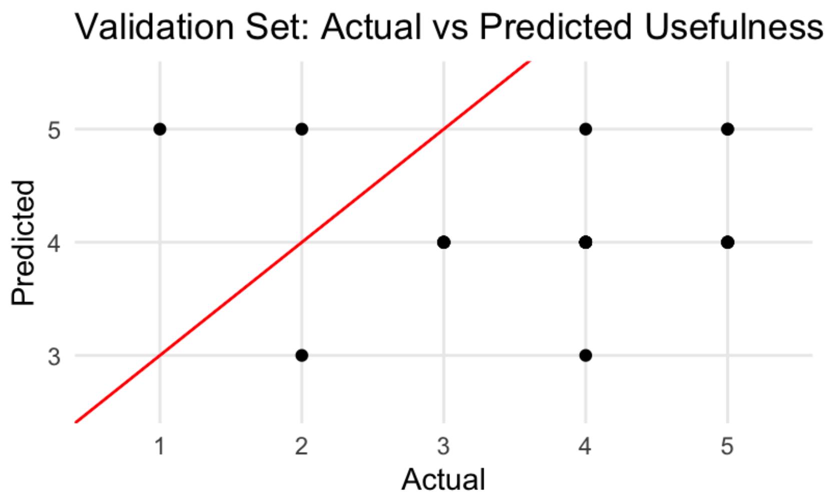
3. Improvements

CODE

```
# Load required libraries
library(readr)
library(dplyr)
library(tidyr)
library(ggplot2)
library(caret)
library(randomForest)
library(e1071)
library(xgboost)

# Load the data files
utterance_train <- read_csv('dialogue_utterance_train.csv')
usefulness_train <- read_csv('dialogue_usefulness_train.csv')
utterance_validation <- read_csv('dialogue_utterance_validation.csv')
usefulness_validation <- read_csv('dialogue_usefulness_validation.csv')

# Standardize column names
colnames(utterance_train) <- c("Dialogue_ID", "Timestamp", "Interlocutor",
"Utterance_text")
colnames(utterance_validation) <- c("Dialogue_ID", "Timestamp", "Interlocutor",
"Utterance_text")

# Function to calculate features
calculate_features <- function(utterance_data, usefulness_data) {
  utterance_data %>%
    group_by(Dialogue_ID) %>%
    summarise(
      Number_of_Utterances = n(),
      Average_Utterance_Length = mean(nchar(Utterance_text), na.rm = TRUE),
      Interlocutor_Ratio = sum(Interlocutor == 'Student') / sum(Interlocutor ==
'Chatbot', na.rm = TRUE)
    ) %>%
    inner_join(usefulness_data, by = "Dialogue_ID")
}

# Calculate features for training and validation sets
train_features <- calculate_features(utterance_train, usefulness_train)
validation_features <- calculate_features(utterance_validation,
usefulness_validation)

# Remove outliers based on IQR
remove_outliers <- function(data) {
  Q1 <- quantile(data$Average_Utterance_Length, 0.25)
  Q3 <- quantile(data$Average_Utterance_Length, 0.75)
  IQR <- Q3 - Q1
  data <- data %>% filter(Average_Utterance_Length >= (Q1 - 1.5 * IQR) &
Average_Utterance_Length <= (Q3 + 1.5 * IQR))
  return(data)
}
train_features <- remove_outliers(train_features)

# Rescale features
preProcValues <- preProcess(train_features[, c("Number_of_Utterances",
"Average_Utterance_Length", "Interlocutor_Ratio")], method = c("center", "scale"))
train_features_scaled <- predict(preProcValues, train_features[,
c("Number_of_Utterances", "Average_Utterance_Length", "Interlocutor_Ratio")])
validation_features_scaled <- predict(preProcValues, validation_features[,
c("Number_of_Utterances", "Average_Utterance_Length", "Interlocutor_Ratio")])

# Add the target variable back to the scaled features
train_features_scaled$Usefulness_score <-
as.factor(train_features$Usefulness_score)
validation_features_scaled$Usefulness_score <-
as.factor(validation_features$Usefulness_score)

# Function to train and evaluate models
train_and_evaluate <- function(model_func, train_data, validation_data) {
  train_x <- train_data[, -4]
  train_y <- train_data$Usefulness_score
  validation_x <- validation_data[, -4]
  validation_y <- validation_data$Usefulness_score

  # Train the model
  model <- model_func(train_x, train_y)

  # Predict on validation set
  validation_pred <- predict(model, validation_x)

  # Calculate evaluation metrics
  cm <- confusionMatrix(validation_pred, validation_y)

  list(model = model, confusion_matrix = cm)
}

# Define models
random_forest_classification <- function(x, y) {
  randomForest(x, y, ntree = 100, importance = TRUE)
}

svm_model <- function(x, y) {
  svm(x, y, probability = TRUE)
}

# Train and evaluate models
rf_results <- train_and_evaluate(random_forest_classification,
train_features_scaled, validation_features_scaled)
svm_results <- train_and_evaluate(svm_model, train_features_scaled,
validation_features_scaled)
```

```
# Print evaluation metrics
cat("Random Forest - Confusion Matrix:\n")
print(rf_results$confusion_matrix)

cat("SVM - Confusion Matrix:\n")
print(svm_results$confusion_matrix)

# Plot predictions vs actual values for the best model
best_model_results <- rf_results
validation_pred <- predict(best_model_results$model, validation_features_scaled[, -
4])

ggplot(data = data.frame(Actual = validation_features_scaled$Usefulness_score,
Predicted = validation_pred), aes(x = Actual, y = Predicted)) +
  geom_point() +
  geom_abline(slope = 1, intercept = 0, color = 'red') +
  labs(title = "Validation Set: Actual vs Predicted Usefulness Scores", x =
"Actual", y = "Predicted") +
  theme_minimal()
```

OUTPUT



Improvements Over Previous Model

□ Outlier Removal: This code includes a function to remove outliers based on the Interquartile Range (IQR) for the average utterance length, which helps in reducing the influence of extreme values on the model.

□ Rescaling Features: Features are rescaled using centering and scaling, ensuring that all variables contribute equally to the model, potentially leading to better performance.

□ Model Variety: This code evaluates two models (Random Forest and SVM) instead of just one, allowing for comparison and selection of the better-performing model.

□ Confusion Matrix: The evaluation metrics include confusion matrices for both models, providing a more detailed performance assessment compared to just RMSE and MAE.

□ Visualization: The code includes a scatter plot to visualize the actual vs predicted usefulness scores, helping to assess model performance visually.

These improvements make the model more robust and allow for a better understanding of its performance, leading to potentially higher accuracy and more reliable predictions.

EXPLANATION

This code builds and evaluates Random Forest and SVM models to predict the usefulness scores of dialogues. As usual we start by loading data and libraries then standardise column names and define the feature calculation feature and proceed to remove outliers which was not previously done. It is removed based on IQR and then we add target variables back to scaled features and train and evaluate the model.Finally, we print evaluation metrics and plot the predicted Vs Actual Values.

4. Dialogue

Below is link to view:
https://docs.google.com/document/d/1c7BXAsvqylNSeGJcrcJPifgUZxofKO0qLBnKagV3woU/edit?usp=sharing

5. Usefulness score

CODE

```
# Load necessary libraries
library(dplyr)
library(readr)
library(randomForest)
library(caret)

# Load the data files
utterance_test <- read_csv('dialogue_utterance_test.csv')
usefulness_test <- read_csv('dialogue_usefulness_test.csv')

# Standardize column names
colnames(utterance_test) <- c("Dialogue_ID", "Timestamp", "Interlocutor",
"Utterance_text")
colnames(usefulness_test) <- c("Dialogue_ID", "Usefulness_score",
"Number_of_Utterances", "Average_Utterance_Length", "Interlocutor_Ratio")

# Function to calculate features
calculate_features <- function(utterance_data) {
  utterance_data %>%
    group_by(Dialogue_ID) %>%
    summarise(
      Number_of_Utterances = n(),
      Average_Utterance_Length = mean(nchar(Utterance_text), na.rm = TRUE),
      Interlocutor_Ratio = sum(Interlocutor == 'Student') / sum(Interlocutor ==
'Chatbot', na.rm = TRUE)
    )
}

# Calculate features for the test set
test_features <- calculate_features(utterance_test)

# Ensure train_features and best_model_results are already defined in the workspace
# Rescale features
preProcValues <- preProcess(train_features[, c("Number_of_Utterances",
"Average_Utterance_Length", "Interlocutor_Ratio")], method = c("center", "scale"))
test_features_scaled <- predict(preProcValues, test_features[,
c("Number_of_Utterances", "Average_Utterance_Length", "Interlocutor_Ratio")])

# Predict using the best-performing model
predicted_scores <- predict(best_model_results$model, test_features_scaled)

# Add predictions to the usefulness_test data frame
```

```
usefulness_test <- usefulness_test %>%
  left_join(test_features, by = "Dialogue_ID") %>%
  mutate(Predicted_Usefulness_Score = predicted_scores)

# Write the results to the existing CSV file
write_csv(usefulness_test, 'dialogue_usefulness_test.csv')

# Verify the changes by reading the file back
predicted_results <- read_csv('dialogue_usefulness_test.csv')
print(head(predicted_results))

# Extract a specific Dialogue_ID for analysis
dialogue_id <- 1849
dialogue_text <- utterance_test %>% filter(Dialogue_ID == dialogue_id) %>%
pull(Utterance_text)
predicted_score <- usefulness_test %>% filter(Dialogue_ID == dialogue_id) %>%
pull(Predicted_Usefulness_Score)

# Print the dialogue and its prediction
cat("Dialogue ID:", dialogue_id, "\n")
cat("Dialogue Text:\n", paste(dialogue_text, collapse = "\n"), "\n")
cat("Predicted Usefulness Score:", predicted_score, "\n")
```

EXPLANATION

First steps are similar till calculating the features. We preprocess and predict the scores and add it to the test data frame.