Assignment 3

Inshal Naqvi

##Question 1

Before you begin, print thefirst few values of the columns with a headercontaining the string“time”.(head(),head())

```
library(dplyr)

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

nycflight = read.csv("flights.csv", sep = ",", header = TRUE)

head(select(nycflight, contains("time")))

##   dep_time sched_dep_time arr_time sched_arr_time air_time   time_hour
## 1      517            515      830            819      227 1/1/13 5:00
## 2      533            529      850            830      227 1/1/13 5:00
## 3      542            540      923            850      160 1/1/13 5:00
## 4      544            545     1004           1022      183 1/1/13 5:00
## 5      554            600      812            837      116 1/1/13 6:00
## 6      554            558      740            728      150 1/1/13 5:00

#summary
```

#Part a

Count the number of flights that departed NYC in the first week (first 7 days) of January and February combined. (filter())

```
library(dplyr)
filtering = filter(nycflight, month < 3 & day < 8)
nrow(filtering)

## [1] 12182
```

##Part b

Print the year, month, day, carrier and air_time of the flights with the 6 longest air times, in descending order of air_time. (select(), arrange())

```
nycflight %>%
  select(year, month, day, carrier, air_time) %>%
  arrange(desc(air_time)) %>%
  head
```

```
##   year month day carrier air_time
## 1 2013     3  17      UA      695
## 2 2013     2   6      HA      691
## 3 2013     3  15      HA      686
## 4 2013     3  17      HA      686
## 5 2013     3  16      HA      683
## 6 2013     2   5      HA      679
```

##Part c

Add a new column to the data frame; speed(in miles per hour) is the ratio of distance to air_time. Note that the unit of speed should be miles per hour. If you think they might be useful, feel free to extract more features than these, and describe what they are. (mutate())

```
wthspeed=
nycflight %>%
  mutate(speed = distance / (air_time/60)) #airtime divided by 60 to convert
                                            # to miles per hour
head(wthspeed)
```

```
##   year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time
## 1 2013     1   1      517            515         2      830            819
## 2 2013     1   1      533            529         4      850            830
## 3 2013     1   1      542            540         2      923            850
## 4 2013     1   1      544            545        -1     1004           1022
## 5 2013     1   1      554            600        -6      812            837
## 6 2013     1   1      554            558        -4      740            728
##   arr_delay carrier flight tailnum origin dest air_time distance hour
minute
## 1        11      UA   1545  N14228    EWR  IAH      227     1400    5
15
## 2        20      UA   1714  N24211    LGA  IAH      227     1416    5
29
## 3        33      AA   1141  N619AA    JFK  MIA      160     1089    5
40
## 4       -18      B6    725  N804JB    JFK  BQN      183     1576    5
45
## 5       -25      DL    461  N668DN    LGA  ATL      116      762    6
0
## 6        12      UA   1696  N39463    EWR  ORD      150      719    5
58
##     time_hour    speed
## 1 1/1/13 5:00 370.0441
```

```
## 2 1/1/13 5:00 374.2731
## 3 1/1/13 5:00 408.3750
## 4 1/1/13 5:00 516.7213
## 5 1/1/13 6:00 394.1379
## 6 1/1/13 5:00 287.6000
```

##Part d

Display the average, min and max air_time times for each month. (group_by(), summarise()). You can exclude NAs for this calculation.

```
na.omit(nycflight) %>%
  group_by(month) %>%
  summarise(avg_airtime = mean(air_time),
            min_airtime = min(air_time),
            max_airtime = max(air_time))
```

```
## # A tibble: 12 × 4
##    month avg_airtime min_airtime max_airtime
##    <int>       <dbl>       <int>       <int>
##  1     1        154.          20         667
##  2     2        151.          21         691
##  3     3        149.          21         695
##  4     4        153.          20         671
##  5     5        146.          21         640
##  6     6        150.          21         650
##  7     7        147.          23         629
##  8     8        148.          21         640
##  9     9        143.          21         636
## 10    10        149.          23         642
## 11    11        155.          24         676
## 12    12        163.          21         661
```

##Part e.1

Impute the missing air_times as the distance divided by the average speed of flights for that destination (dest). Make a second copy of your dataframe, but this time impute missing air_time with the average air_time for that destination. What assumptions do these data filling methods make? Which is the bestway to impute the data, or do you see a better way, and why? You may impute or remove other variables as you find appropriate. Briefly explain your decisions.(group_by(),mutate())

```
wthspeed1=
  nycflight %>%
  mutate(speed = distance / (air_time/60))

newflights1=
wthspeed1 %>%

  group_by(dest) %>%
```

```
  mutate(air_time=ifelse(is.na(air_time), distance / mean(na.omit(speed)),
air_time)) %>%

           select(air_time, dest, speed)
head(newflights1)
```

```
## # A tibble: 6 × 3
## # Groups:   dest [5]
##   air_time dest  speed
##      <dbl> <chr> <dbl>
## 1      227 IAH    370.
## 2      227 IAH    374.
## 3      160 MIA    408.
## 4      183 BQN    517.
## 5      116 ATL    394.
## 6      150 ORD    288.
```

##Part e.2

```
wthspeed2=

nycflight %>%
  group_by(dest) %>%
    mutate(speed = distance / (air_time/60)) %>%
      mutate(air_time=ifelse(is.na(air_time),
mean(na.omit(air_time)),air_time)) %>%
        select(air_time, dest, speed)
head(wthspeed2)
```

```
## # A tibble: 6 × 3
## # Groups:   dest [5]
##   air_time dest  speed
##      <dbl> <chr> <dbl>
## 1      227 IAH    370.
## 2      227 IAH    374.
## 3      160 MIA    408.
## 4      183 BQN    517.
## 5      116 ATL    394.
## 6      150 ORD    288.
```

The best way in my opinion is by replacing values with mean since that wont have any affect when statistical operation is applied on the value.

##Question2

```
library(tidyr)
who1<- tidyr::who
who1
```

```
## # A tibble: 7,240 × 60
##    country     iso2  iso3   year new_sp_m014 new_sp_m1524 new_sp_m2534
new_sp_m3544
##    <chr>       <chr> <chr> <int>       <int>        <int>        <int>
<int>
##  1 Afghanistan AF    AFG    1980          NA           NA           NA
NA
##  2 Afghanistan AF    AFG    1981          NA           NA           NA
NA
##  3 Afghanistan AF    AFG    1982          NA           NA           NA
NA
##  4 Afghanistan AF    AFG    1983          NA           NA           NA
NA
##  5 Afghanistan AF    AFG    1984          NA           NA           NA
NA
##  6 Afghanistan AF    AFG    1985          NA           NA           NA
NA
##  7 Afghanistan AF    AFG    1986          NA           NA           NA
NA
##  8 Afghanistan AF    AFG    1987          NA           NA           NA
NA
##  9 Afghanistan AF    AFG    1988          NA           NA           NA
NA
## 10 Afghanistan AF    AFG    1989          NA           NA           NA
NA
## # … with 7,230 more rows, and 52 more variables: new_sp_m4554 <int>,
## #   new_sp_m5564 <int>, new_sp_m65 <int>, new_sp_f014 <int>,
## #   new_sp_f1524 <int>, new_sp_f2534 <int>, new_sp_f3544 <int>,
## #   new_sp_f4554 <int>, new_sp_f5564 <int>, new_sp_f65 <int>,
## #   new_sn_m014 <int>, new_sn_m1524 <int>, new_sn_m2534 <int>,
## #   new_sn_m3544 <int>, new_sn_m4554 <int>, new_sn_m5564 <int>,
## #   new_sn_m65 <int>, new_sn_f014 <int>, new_sn_f1524 <int>, …
```

##Part a Explain why this line mutate(key=stringr::str_replace(key,“newrel”,“new_rel”)) is necessary to properly tidy the data. What happens if you skip this line?

#answer The column “newrel” makes the dataframe inconsistant hence to maintain consistance with the format “newrel” is replaced with “new_rel”. This is used later on to extract information in a much cleaner way.

##Part b How many entries are removed from the dataset when you set values_drop_na to true in the pivot_longer command (in this dataset)?

#answer First lets check how many entries were there initially.

```
who1 = who %>%
  pivot_longer(
    col = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases"
```

```
  )
head(who1)
```

```
## # A tibble: 6 × 6
##   country     iso2  iso3   year key          cases
##   <chr>       <chr> <chr> <int> <chr>        <int>
## 1 Afghanistan AF    AFG    1980 new_sp_m014     NA
## 2 Afghanistan AF    AFG    1980 new_sp_m1524    NA
## 3 Afghanistan AF    AFG    1980 new_sp_m2534    NA
## 4 Afghanistan AF    AFG    1980 new_sp_m3544    NA
## 5 Afghanistan AF    AFG    1980 new_sp_m4554    NA
## 6 Afghanistan AF    AFG    1980 new_sp_m5564    NA
```

Here we see that there are 405,440 rows in total. Now lets run pivot_longer and drop the NA values.

```
who1 = who %>%
  pivot_longer(
    col = new_sp_m014:newrel_f65,
    names_to = "key",
    values_to = "cases",
    values_drop_na = TRUE
  )
head(who1)
```

```
## # A tibble: 6 × 6
##   country     iso2  iso3   year key          cases
##   <chr>       <chr> <chr> <int> <chr>        <int>
## 1 Afghanistan AF    AFG    1997 new_sp_m014      0
## 2 Afghanistan AF    AFG    1997 new_sp_m1524    10
## 3 Afghanistan AF    AFG    1997 new_sp_m2534     6
## 4 Afghanistan AF    AFG    1997 new_sp_m3544     3
## 5 Afghanistan AF    AFG    1997 new_sp_m4554     5
## 6 Afghanistan AF    AFG    1997 new_sp_m5564     2
```

Now we see we that the number of rows are 76,046. This means that rows with NA 329,394

##Part c Explain the difference between an explicit and implicit missing value, in general. Can you find any implicit missing values in this dataset, if so where?

#answer Explicit missing values are where the meaning is clearly defined in the place where there is suppose to be a value. Like when there is “NA” wehre there should be a value. With implicit missing values, it is not clearly defined. implicit missing values do not give any clear information. With this data set we dont see any implicitly missing values. One could argue that 0s might be implicitly missing values but in this data set it just means there were no TB cases.

##Part d Looking at the features (country, year, var, sex, age, cases) in the tidied data, are they all appropriately typed? Are there any features you think would be better suited as a different type? Why or why not?

#answer The year column seems to be good and convey appropriate imformation that it should. However the rest could be represented better by converting them into factor type. Like age should be an int instead of chr.

##Part e Generatean informative visualization, which shows something about the data. Give a brief description of what it shows, and why you thought it would be interesting to investigate.

#Answer With this we can probably see the speard of Tb with a certain country and check if one variant is spreading more in a certain country

```
library(tidyr)
who=tidyr::who
who1 <- who %>%
  gather(new_sp_m014:newrel_f65, key = "key", value = "cases", na.rm = TRUE)
who2 <- who1 %>%
mutate(key = stringr::str_replace(key, "newrel", "new_rel"))

who3 <- who2 %>%
separate(key, c("new", "type", "sexage"), sep = "_")
who4 <- who3 %>% select(-new, -iso2, -iso3)
who5 <- who4 %>%
separate(sexage, c("sex", "age"), sep = 1)
who_sp= who5%>% spread(key=country,value=cases)

who_bhutan=
who_sp%>% select(Bhutan,type,year)
Bhutan_TBcases=na.omit(who_bhutan)
head(Bhutan_TBcases)
```

```
## # A tibble: 6 × 3
##   Bhutan type   year
##    <int> <chr> <int>
## 1     12 sp     1995
## 2     43 sp     1995
## 3     44 sp     1995
## 4     25 sp     1995
## 5     12 sp     1995
## 6      9 sp     1995
```

##Part f Lets construct the table

```
Group = c(1, 1, 1, 1, 2, 2, 2, 2, 3, 3, 3, 3)
Year = c(2006, 2007, 2008, 2009, 2006, 2007, 2008, 2009, 2006, 2007, 2008,
2009)
Qrt.1 = c(15, 12, 22, 10, 12 ,16, 13, 23, 11, 13, 17, 14)
Qrt.2 = c(16, 13, 22, 14, 13, 14, 11, 20,12, 11, 12, 9)
Qrt.3 = c(19, 27, 24, 20, 25, 21, 29, 26, 22, 27, 23, 31)
Qrt.4 = c(17, 23, 20, 16, 18 ,19 , 15, 20, 16, 21, 19, 24)
```

```
qrtRev = data.frame(Group, Year, Qrt.1, Qrt.2, Qrt.3, Qrt.4)

head(qrtRev)
```

```
##   Group Year Qrt.1 Qrt.2 Qrt.3 Qrt.4
## 1     1 2006    15    16    19    17
## 2     1 2007    12    13    27    23
## 3     1 2008    22    22    24    20
## 4     1 2009    10    14    20    16
## 5     2 2006    12    13    25    18
## 6     2 2007    16    14    21    19
```

Lets tidy it up now

```
new_qrtRev <- qrtRev %>% gather(Quarter, Revenue, Qrt.1:Qrt.4)
head(new_qrtRev)
```

```
##   Group Year Quarter Revenue
## 1     1 2006   Qrt.1      15
## 2     1 2007   Qrt.1      12
## 3     1 2008   Qrt.1      22
## 4     1 2009   Qrt.1      10
## 5     2 2006   Qrt.1      12
## 6     2 2007   Qrt.1      16
```

```
final_qrtRev = new_qrtRev %>% separate(Quarter, c("Time_Interval",
"Interval_ID"))
head(final_qrtRev)
```

```
##   Group Year Time_Interval Interval_ID Revenue
## 1     1 2006           Qrt           1      15
## 2     1 2007           Qrt           1      12
## 3     1 2008           Qrt           1      22
## 4     1 2009           Qrt           1      10
## 5     2 2006           Qrt           1      12
## 6     2 2007           Qrt           1      16
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.