Assignment3
Jinyang Ruan
011696096

1.0 Load the data into R and print the first few values of the columns with a header containing the string “time”.

```{r}
flights <- read.csv("E:/WSU Graduate/CPT_S 575 Data Science/flights.csv", header = TRUE)
```

```{r 1.0}
flight_time <- subset(flights[c(grep(pattern = "time", colnames(flights)))])
head(flight_time, n=10)
```

Description: df[,6] [10 x 6]

	dep_time <int>	sched_dep_time <int>	arr_time <int>	sched_arr_time <int>	air_time <int>	time_hour <chr>
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
7	555	600	913	854	158	1/1/13 6:00
8	557	600	709	723	53	1/1/13 6:00
9	557	600	838	846	140	1/1/13 6:00
10	558	600	753	745	138	1/1/13 6:00

1-10 of 10 rows

In this case, I printed first 10 rows with a header containing the string “time”.

1.a (10 pts) Count the number of flights that departed NYC in the first week (first 7 days) of January and February combined.

```{r 1.a}
##Count the number of flights that departed NYC in the first week (first 7 days) of January and February combined.

flights_1a <- filter(flights, month == 1 | month == 2, day <= 7)

count(flights_1a)
```

Description: df[,1] [1 x 1]

n <int>
12182

1 row

1.b (10 pts) Print the year, month, day, carrier and air_time of the flights with the 6 longest air times, in descending order of air_time.

```{r 1.b}
##Print the year, month, day, carrier and air_time of the flights with the 6 longest airtimes, in descending order of air_time.

flights_1b <- arrange(flights, desc(air_time))
flights_1b_6th <- flights_1b[1:6,]
select(flights_1b_6th, year, month, day, carrier, air_time)
```

Description: df[,5] [6 x 5]

	year <int>	month <int>	day <int>	carrier <chr>	air_time <int>
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

6 rows

1.c (10 pts) Add a new column to the dataframe; 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.

```
flights <- flights %>% mutate(speed = distance / air_time * 60)
head(flights, n=6)
```

```
##   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
```

1.d (14 pts) Display the average, min and max air_time times for each month.

I excluded NAs first then computed average, min and max air_times for each month.

```
flights_1d <- subset(flights, air_time != "NA")
flights_1d_1 = flights_1d %>%
               group_by(month) %>%
               summarise(average = mean(air_time), min = min(air_time), max = max(air_time))
print(flights_1d_1)
```

```
## # A tibble: 12 x 4
##    month average   min   max
##    <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
```

1.e (16 pts) 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 best way 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.

Firstly, I imputed the missing air_times as the distance divided by the average speed of flights for that destination.

```
# Impute the missing air_times as the distance divided by the average speed of flights
flights_1e_1 = flights%>%
  group_by(dest)%>%
  select("dest", "distance", "air_time")%>%
  mutate(air_time=ifelse(is.na(air_time),
                      ifelse(is.nan(mean(air_time, na.rm = TRUE)),0,
                   distance / mean(distance, na.rm =TRUE) * mean(air_time, na.rm = TRUE)),
                      air_time))%>%
  ungroup(dest)
head(flights_1e_1)
```

```
## # A tibble: 6 x 3
##   dest  distance air_time
##   <chr>    <int>    <dbl>
## 1 IAH       1400      227
## 2 IAH       1416      227
## 3 MIA       1089      160
## 4 BQN       1576      183
## 5 ATL        762      116
## 6 ORD        719      150
```

Then I imputed missing air_time with the average air_time for that destination.

```
# Impute the missing air_times with the average air_time
flights_1e_2 = flights%>%
  group_by(dest)%>%
  select( "dest", "distance", "air_time")%>%
  mutate(air_time=ifelse(is.na(air_time),
                      ifelse(is.nan(mean(air_time,na.rm = TRUE)),0,
```

```
                mean(air_time,na.rm = TRUE)),
                      air_time))%>%
    ungroup(dest)
head(flights_1e_2)
```

```
## # A tibble: 6 x 3
##   dest  distance air_time
##   <chr>    <int>    <dbl>
## 1 IAH       1400      227
## 2 IAH       1416      227
## 3 MIA       1089      160
## 4 BQN       1576      183
## 5 ATL        762      116
## 6 ORD        719      150
```

When we impute the missing air_times as the distance divided by the average speed of flights for that destination, we assume that for the same destination, the average speeds of the flights are

similar; while choosing the average air_time, the air_time of the samples are similar. Here imputation by distance divided by average speed is a better way for the air_time which usually associates with the distance.
However, usually this "Mean Imputation" is not a good choice in practice. there are many other better solutions, such as "Hot deck imputation", "Regression imputation", and "Stochastic regression imputation". In this case, to impute missing air_time. The first way which is the distance divided by the average speed of flights for that destination would be good.

2.0 Lode the dataset into R and tidy the dataset

````
```{r load the dataset into R and tidy the dataset}

who = tidyr::who
who1 = who %>%
      pivot_longer(cols = new_sp_m014:newrel_f65, names_to = "key", values_to = "cases", values_drop_na = FALSE)%>%
      mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
      separate(key, c("new", "var", "sexage")) %>%
      select(-new, -iso2, -iso3) %>%
      separate(sexage, c("sex", "age"), sep = 1)
who1
```
````

````
## # A tibble: 405,440 x 6
##    country      year var   sex   age   cases
##    <chr>       <int> <chr> <chr> <chr> <int>
##  1 Afghanistan  1980 sp    m     014      NA
##  2 Afghanistan  1980 sp    m     1524     NA
##  3 Afghanistan  1980 sp    m     2534     NA
##  4 Afghanistan  1980 sp    m     3544     NA
##  5 Afghanistan  1980 sp    m     4554     NA
##  6 Afghanistan  1980 sp    m     5564     NA
##  7 Afghanistan  1980 sp    m     65       NA
##  8 Afghanistan  1980 sp    f     014      NA
##  9 Afghanistan  1980 sp    f     1524     NA
## 10 Afghanistan  1980 sp    f     2534     NA
## # ... with 405,430 more rows
````

2.a
It replaces the header with name "newrel" to "new_rel", which can make the format of all headers consistent.
If it is skipped, we cannot extract detailed features by simply separate the header name.
For example, given a header name of "new_sp_f5564" we can apply separate function on it and extract three new features:new, sp, f5564, while in case of "newrel_m014" we cannot.

2.b (5 pts) How many entries are removed from the dataset when you set values_drop_na to true in the pivot_longer command (in this dataset)?

````
```{r 2.b}
who2 = who %>%
      pivot_longer(cols = new_sp_m014:newrel_f65, names_to = "key", values_to = "cases", values_drop_na = TRUE)%>%
      mutate(key = stringr::str_replace(key, "newrel", "new_rel")) %>%
      separate(key, c("new", "var", "sexage")) %>%
      select(-new, -iso2, -iso3) %>%
      separate(sexage, c("sex", "age"), sep = 1)
who2
# when set values_drop_na=TRUE, the remove entries can be computed as:
count(who1) - count(who2)
```
````

There is the data frame for who2.

tbl_df 76046 x 6 | data.frame 1 x 1

A tibble: 76,046 x 6

country <chr>	year <int>	var <chr>	sex <chr>	age <chr>	cases <int>
Afghanistan	1997	sp	m	014	0
Afghanistan	1997	sp	m	1524	10
Afghanistan	1997	sp	m	2534	6
Afghanistan	1997	sp	m	3544	3
Afghanistan	1997	sp	m	4554	5
Afghanistan	1997	sp	m	5564	2
Afghanistan	1997	sp	m	65	0
Afghanistan	1997	sp	f	014	5
Afghanistan	1997	sp	f	1524	38
Afghanistan	1997	sp	f	2534	36

1-10 of 76,046 rows Previous 1 2 3 4 5 6 ... 100 Next

The removed entries can be computed as following:

```
# When set values_drop_na=TRUE, the remove entries can be computed as:
count(who1) - count(who2)
```

```
##         n
## 1 329394
```

2.c(5 pts) 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?

According to the "R for Data Science, An explicit missing value is the presence of an absence; an implicit missing value is the absence of a presence", That means for explicit missing, there will be a specific representation to indicate the missing of the value (e.g., NA). While for implicit missing, there will be no specific representation for the value.

In this dataset, we can consider the “case == 0” as the implicit missing value, we can get the sub-samples with following command.

```{r}
who %>%
pivot_longer(cols = new_sp_m014:newrel_f65, names_to = "key", values_to = "cases", values_drop_na = TRUE) %>%
filter(cases == 0)
```

A tibble: 11,080 x 6

country <chr>	iso2 <chr>	iso3 <chr>	year <int>	key <chr>	cases <int>
Afghanistan	AF	AFG	1997	new_sp_m014	0
Afghanistan	AF	AFG	1997	new_sp_m65	0
Afghanistan	AF	AFG	1997	new_sp_f5564	0
Afghanistan	AF	AFG	2007	new_sn_m014	0
Afghanistan	AF	AFG	2007	new_sn_m1524	0
Afghanistan	AF	AFG	2007	new_sn_m2534	0
Afghanistan	AF	AFG	2007	new_sn_m3544	0
Afghanistan	AF	AFG	2007	new_sn_m4554	0
Afghanistan	AF	AFG	2007	new_sn_m5564	0
Afghanistan	AF	AFG	2007	new_sn_m65	0

1-10 of 11,080 rows Previous 1 2 3 4 5 6 ... 100 Next

2.d (5 pts) 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?

```
head(as_tibble(who2))
```

```
## # A tibble: 6 x 6
##   country      year var   sex   age   cases
##   <chr>       <int> <chr> <chr> <chr> <int>
## 1 Afghanistan  1997 sp    m     014       0
## 2 Afghanistan  1997 sp    m     1524     10
## 3 Afghanistan  1997 sp    m     2534      6
## 4 Afghanistan  1997 sp    m     3544      3
## 5 Afghanistan  1997 sp    m     4554      5
## 6 Afghanistan  1997 sp    m     5564      2
```

```
sapply(who2, class)
```

```
##     country        year         var         sex         age       cases
## "character"   "integer" "character" "character" "character"   "integer"
```

As shown above, all features are typed appropriately except the age which is typed as character. Usually, it is better to type this feature as integer, when we try to analyze this feature (e.g., distribution of the age, average age), we must compute the results with integer.

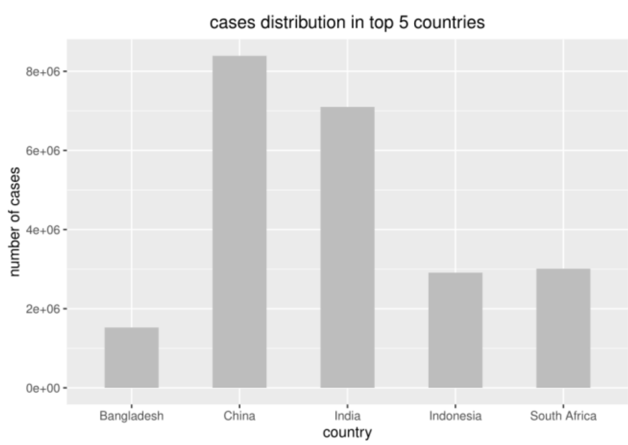
2.e (10 pts) Generate an informative visualization, which shows something about the data. Give a brief description of what it shows, and why you thought it would be interesting to investigate.

I. Look at the data grouped by country.

```
##look at the data grouped by country
VDCountry = who2%>%
            group_by(country)%>%
            tally(cases)%>%
            top_n(5)
```

```
## Selecting by n
```

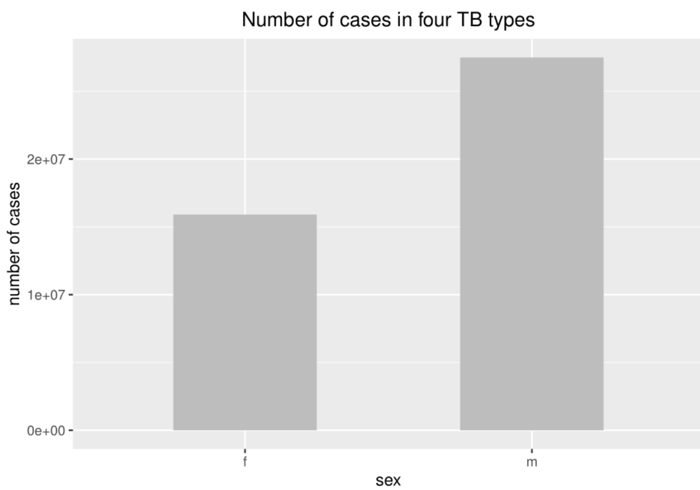
```
ggplot(data=VDCountry, aes(x=country, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="cases distribution in top 5 countries",
            x="country", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



In this case, top TB cases distribution by countries are shown as following graph, from which we can know which countries are worse off.

II. look at the data grouped by sex

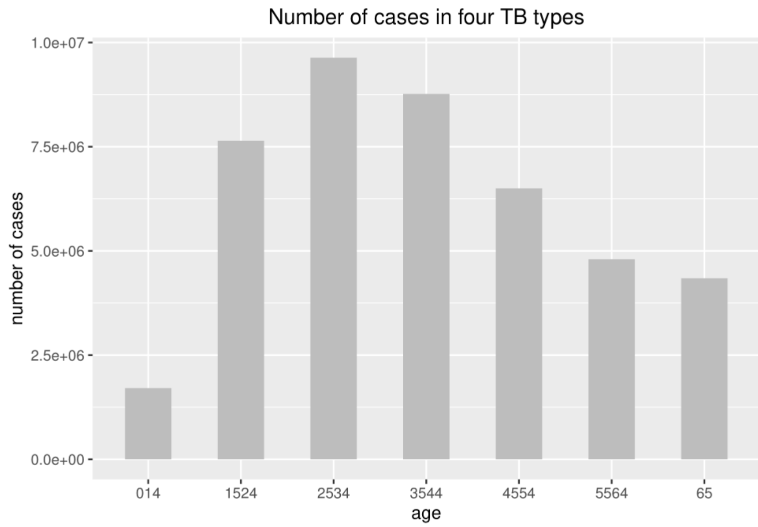
```
## look at the data grouped by sex

VDSex = who2%>%
        group_by(sex)%>%
        tally(cases)
ggplot(data=VDSex, aes(x=sex, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="Number of cases in four TB types",
            x="sex", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



the above graph shows TB case distribution among female and male, it indicates that male are more easily affected.

III. Look at the data grouped by age.

```
## Look at the data grouped by age
VDAge = who2%>%
        group_by(age)%>%
        tally(cases)
ggplot(data=VDAge, aes(x=age, y=n)) +
       geom_bar(stat="identity",width=0.5,fill="gray") +
       labs(title="Number of cases in four TB types",
            x="age", y="number of cases") +
       theme(plot.title = element_text(hjust = 0.5))
```



the above graph shows TB case distribution among different ages, it indicates that people with young and the middle-aged are more possible to be affected.

2.f

Firstly, I built the data frame and input all data.

```
qtrRev = tibble(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),
                Qtr.1=c(15,12,22,10,12,16,13,23,11,13,17,14),
                Qtr.2=c(16,13,22,14,13,14,11,20,12,11,12,9),
                Qtr.3=c(19,27,24,20,25,21,29,26,22,27,23,31),
                Qtr.4=c(17,23,20,16,18,19,15,20,16,21,19,24))
print(qtrRev)
```

```
## # A tibble: 12 x 6
##    Group  Year Qtr.1 Qtr.2 Qtr.3 Qtr.4
##    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  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
##  7     2  2008    13    11    29    15
##  8     2  2009    23    20    26    20
##  9     3  2006    11    12    22    16
## 10     3  2007    13    11    27    21
## 11     3  2008    17    12    23    19
## 12     3  2009    14     9    31    24
```

Then I tidied the data as required.

```{r}

qtrRev_Tidy = qtrRev %>%
  pivot_longer(cols = Qtr.1:Qtr.4, names_to = "Interval_ID", values_to = "Revenue", values_drop_na = FALSE) %>%
  separate(Interval_ID, c("Time_Interval","Interval_ID"))
qtrRev_Tidy

```

```
## # A tibble: 48 x 5
##    Group  Year Time_Interval Interval_ID Revenue
##    <dbl> <dbl> <chr>         <chr>         <dbl>
##  1     1  2006 Qtr           1                15
##  2     1  2006 Qtr           2                16
##  3     1  2006 Qtr           3                19
##  4     1  2006 Qtr           4                17
##  5     1  2007 Qtr           1                12
##  6     1  2007 Qtr           2                13
##  7     1  2007 Qtr           3                27
##  8     1  2007 Qtr           4                23
##  9     1  2008 Qtr           1                22
## 10     1  2008 Qtr           2                22
## # ... with 38 more rows
```