Assignment

Ammara Tayyab

2022-07-24

Loading libraries

```
library(data.table)
library(tidyverse)
library(ggplot2)
library(flexdashboard)
library(readxl)
library(xlsx)
library(kableExtra)
```

Question 1

reading data from excel files

```
df1  <- read_excel("~/Downloads/r projects from freelanc/ass24/Counties Data Set 1.xlsx")
df2  <- read_excel("~/Downloads/r projects from freelanc/ass24/Counties Data Set 2.xlsx")
df3  <- read_excel("~/Downloads/r projects from freelanc/ass24/Counties Data Set 3.xlsx")
```

making identical column names to combine 3 data sets

```
names(df2) <- c("county",      "state"      , "pop.density" ,"pop"      ,
   "pop.change" , "age6574"    ,  "age75"    ,    "crime"    ,    "college" ,    "income"      ,
"farm" ,       "democrat" ,   "republican",  "white"   ,     "black"     ,   "turnout"  )

names(df3) <- c("county",      "state"      , "pop.density" ,"pop"      ,
    "pop.change" , "age6574"    ,  "age75"    ,    "crime"    ,    "college" ,    "income"      ,
"farm" ,       "democrat" ,   "republican",  "white"   ,     "black"     ,   "turnout"  )
```

Combining the 3 data sets row wise so that no data gets lost

```
df <- rbind(df1,df2,df3)
```

Question 2 "Dealing with missing values and calculating percentage of missing in 16 variables"

```
map(df, ~mean(is.na(.))*100)
```

$county [1] 0

$state [1] 0

$pop.density [1] 0

$pop [1] 0

$pop.change [1] 0

$age6574 [1] 0

$age75 [1] 0.09551098

$crime [1] 0.127348

$college [1] 0

$income [1] 0.127348

$farm [1] 0

$democrat [1] 0.8595989

$republican [1] 0.8595989

$white [1] 0

$black [1] 0

$turnout [1] 0.09551098 The above results shows that overall there are less than 5% of missing values in the data set with highest percentage for democrat column.

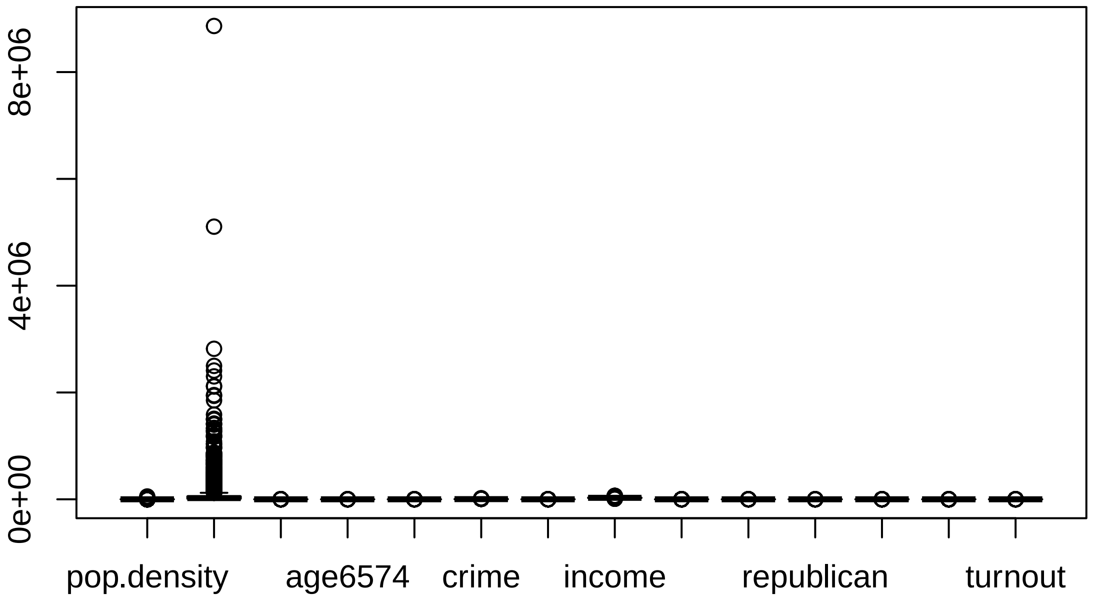
Missing values in R are dealt by using `is.na` or `na.omit` function. `na.omit` removes all rows with `nan` values. We can calculate the sum of missing values in each column and then divide it by the total number of rows to calculate percentage. For an easy solution we use base R map function below. Usually for the numerical variables we can fill missing values with mean value if we do not want to lose large number of rows and with mode for categorical variables.

Question 3 "Identify if there are any outliers in the data set"

An outlier is a point which is distant from the other majority points in the data. Usually they are small in number and we can identify outliers by boxplot which shows the median and quartile of the data set. Another graphical illustration is histogram which can help us to check outliers. Other methods to detect outliers are using z-score and other statistical methods (Grubb, Dixon and Rosner tests).

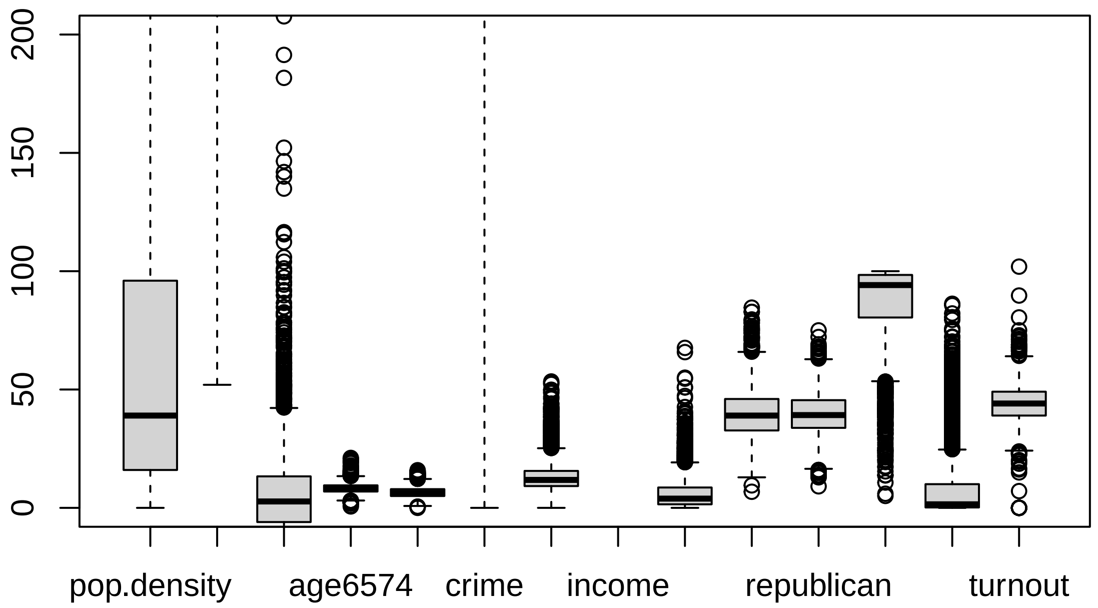
As an example we plot boxplot for all numerical variables in whole datset.

```
boxplot(df[,3:16])
```



The above graph shows that variables population density has highest number of outliers as compared to other variables. Lets chnage the ylimit of the graph to see the outliers more clearly.

```
boxplot(df[,3:16], ylim = c(0,200))
```



By having a closer we can observe that outliers are present in each column. We can actually extract outlier values from the each data column. For example lets show the outlier values in Income variable.

```
boxplot.stats(df$income)$out
```

```
[1] 47188 50098 57100 44141 54088 48339 51716 47600 45252 53167 49327 43693 [13] 52929 49083 45037 51651
59157 51167 53430 53670 50091 44874 43782 54244 [25] 44679 52976 57990 48008 49061 50891 46058 51436
45216 48415 53845 48000 [37] 45313 54920 45457 46687 52308 47911 45510 51167 44257 50348 11110 11362
[49] 49706 44502 50845 46491 49724 45847 45923 61088 61988 48471 45794 52112 [61] 49209 54915 46872
49910 44586 50980 47308 43554 47136 44189 43890 48098 [73] 44571 44634 45214 46249 46942 57640 47641
44216 61132 48490 51835 53590 [85] 62749 62255 53040 48862 45770 60798 49305 60619 44039 58892 50664
60479 [97] 53247 58862 44302 45158 48851 52325 44323 51353 44050 11502 48332 52987 [109] 44945 46163
47397 10903 43972 55346 48064 65201 48806 45517 45283 56006 [121] 52078 43596 47526 50812 56419 62187
51045 48186 44555 47578 49096
```

The above answer the question that there are outliers in our data. Now we will move towards removing outliers. We will use Z-score method.

```
df1  <- df[,3:16]
df1 <-  na.omit(df1)
z_scores <- as.data.frame(sapply(df1, function(df1) (abs(df1-mean(df1))/sd(df1))))
```

only keeping rows with z-score less than 3

```
no_outliers <- df1[!rowSums(z_scores>3), ]

no_outliers <- data.frame(no_outliers)

head(no_outliers) %>% kbl(.,booktabs=T,digits = c(2, 2, 2,2,2),format="latex",align="c") %>%
     kable_styling(full_width = T,latex_options="HOLD_position",font_size=6)
```

pop.density	pop	pop.change	age6574	age75	crime	college	income	farm	democrat	republican	white	black	turnout
61	34222	11.9	5.7	4.1	4996	14.5	32240	1.8	30.9	55.9	79.32	20.00	45.54
67	98280	35.4	9.2	6.0	3329	16.8	30199	1.7	26.2	56.5	86.04	12.86	47.29
29	25417	2.0	8.2	6.4	3192	11.8	23838	2.4	46.4	42.9	55.55	44.04	41.04
28	16576	9.2	6.7	6.0	0	4.7	23714	0.9	43.2	46.5	78.74	20.98	40.54
62	39248	10.6	7.4	5.6	2052	7.0	26323	4.7	32.9	53.8	97.83	1.33	42.05
28	21892	0.1	8.8	8.0	3530	8.0	21499	3.5	47.4	41.2	59.61	40.19	38.73

```
dim(no_outliers)
```

```
[1] 2705 14
```

```
dim(df1)
```

```
[1] 3102 14
```

After removing outliers we can see that the data set is smaller with 2705 rows.

question 4

Function to calculate mean,median,Sd, min and max of variables

```
myfunc  <- function(df1,cols=names(df1)){

combined =data.frame()

for(i in 1:length(cols)){

    mean=mean(df1[,cols[i]])
    median = median(df1[,cols[i]])
    sd=sd(df1[,cols[i]])
    minimum= min(df1[,cols[i]])
    maximum = max(df1[,cols[i]])
    com  <- c(cols[i],mean,median,sd,minimum,maximum)
    #cat(com)
    combined= rbind(combined,com)
}

    return(kbl(combined,booktabs=T,digits = c(0, 0, 2,2,2),format="latex",
    col.names=c("Variable","Mean","Median","STD","Minimum","Maximum"),align="c") %>%
     kable_styling(full_width = T,latex_options="HOLD_position",font_size=8))
    }
```

calling the function to get a table of 5 values for each variable

```
myfunc(no_outliers,c("pop.density", "pop.change", "age6574", "age75",
 "crime", "college", "income", "farm", "democrat",
 "republican", "white", "black", "turnout"))
```

Variable	Mean	Median	STD	Minimum	Maximum
pop.density	114.474676524954	39	288.378997495487	0	3914
pop.change	4.62314231873881	2.40000009536743	15.5001024762858	- 34.4000015258789	65.5
age6574	8.36110905516831	8.30000019073486	1.87607450322013	2.20000004768372	14.8000001907349
age75	6.69759704500382	6.40000009536743	2.19228360296005	0.800000011920929	13.6000003814697
crime	2850.07356746765	2580	2036.84064315462	0	9930
college	12.6836968637879	11.6000003814697	5.10838775565028	3.70000004768372	33
income	27983.9290203327	27312	5766.39110436335	11110	49305
farm	6.38510167035455	4.30000019073486	6.22796143834753	0	28.5
democrat	39.1059889016002	38.7999992370605	9.71700159338596	9.5	72
republican	40.1075785578729	39.4000015258789	8.14590448814265	15.8000001907349	65.3000030517578
white	89.2033313349303	95.0113525390625	12.5318608954412	42.1552314758301	99.9481353759766
black	7.26297810459492	1.28216791152954	11.7648311764736	0	51.663875579834
turnout	43.9010336602682	44.1430130004883	6.8971570146921	22.2883224487305	66.8500671386719

Question6 "Discuss what is parallel programming and how it can be performed in R. Use parallel programming to run your function to perform Exploratory Data Analysis on the given dataset. Report how parallel

programming helped speed up your code.”

Parallel programming is a method of running multiple tasks in parallel. It is used to speed up the code by running multiple tasks in parallel for a PC with multiple cores. R is not a parallel programming language but it can be used to run multiple tasks in parallel with the help of packages. R like other langauges such as python, julia is a interpreted language unlike C,C++ which are compiled languages and they are faster than interpreted languages. Parallel programming saves time and uses all the avaiable cores of the PC. One more advantage of parallel programming is that it can use all idle cores of PC as well which are not usually used even if RAM memory is full. For Exploratory Data Analysis we can use parallel programming to speed up the code since it requires lots of data analysis with plots such density plots, corelation plots etc. It can help in machine learning to train the model in less time. Due to these popular attributes C++ is mostly used for high performance tasks as compared to R. Parallel computing in R usually works by two methods local parallelism and distributed parallelism/clustering. Local parallelism is used to run multiple tasks in parallel on a single machine. Distributed parallelism is used to run multiple tasks in parallel on multiple machines remotely.

We will use some packages of R which have the option to run tasks in parallel. One popular example is the Parallel package. In the end we will compare times of the tasks with and without parallel computing for the same data set.

Exploratory Data Analysis

loading libraries required

```
library(doParallel)
```

```
## Loading required package: foreach
```

```
##
## Attaching package: 'foreach'
```

```
## The following objects are masked from 'package:purrr':
##
##     accumulate, when
```

```
## Loading required package: iterators
```

```
## Loading required package: parallel
```

```
library(foreach)
library(parallel)
```

1. Measuring time for mean computation on single core with different plots and analysis

```
system.time(delay<-df %>%
              group_by(income) %>%
              summarise(
                count=n(),
                dist=mean(age75, na.rm=TRUE),
                delay=mean(farm, na.rm=TRUE)) %>%
              filter(count>20, dist<2000, !is.na(delay)) %>%
              collect())
```

user system elapsed 0.111 0.003 0.116

```
system.time(cor.test(df$democrat,df$republican, na.rm=TRUE))
```

user system elapsed 0.001 0.000 0.001

```
system.time(lm(df$democrat~df$republican, na.rm=TRUE))
```

```
## Warning: In lm.fit(x, y, offset = offset, singular.ok = singular.ok, ...) :
```

```
##  extra argument 'na.rm' will be disregarded
```

user system elapsed 0.003 0.000 0.004

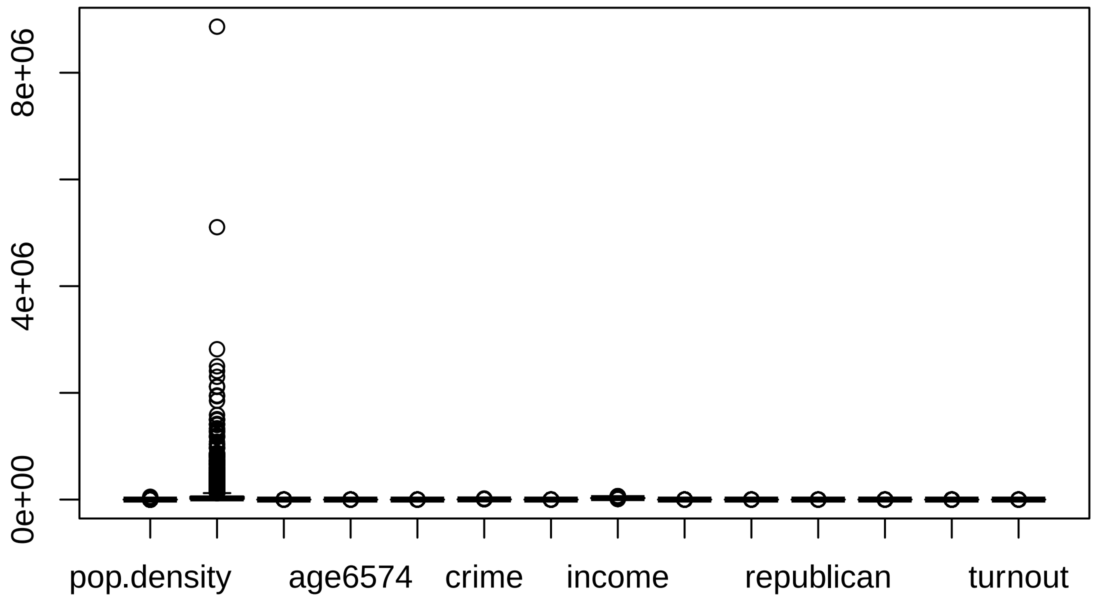
```
system.time(View(df))
```

user system elapsed 0.137 0.038 3.880

```
system.time(ggplot(data=df,aes(x=reorder(county,pop.density),y=income)) +
  geom_bar(stat ='identity',aes(fill=income))+
  coord_flip() +
  theme_grey() +
  scale_fill_gradient(name="Maths Score Level")+
  labs(title = 'income and population relation',
       y='population density',x='County')+
  geom_hline(yintercept = mean(df$income),size = 1, color = 'blue'))
```

user system elapsed 0.023 0.002 0.025

```
system.time(boxplot(df[,3:16]))
```



user system elapsed 0.012 0.000 0.011

```
ggplot(data = df, aes(x=crime,y=state, color=county)) +
  geom_boxplot()+
  scale_color_brewer(palette="Dark2") +
  geom_jitter(shape=16, position=position_jitter(0.2))+
  labs(title = 'is there crime in the county',
       y='state',x='crime')
```

```
## Warning: Removed 4 rows containing non-finite values (stat_boxplot).

## Warning in RColorBrewer::brewer.pal(n, pal): n too large, allowed maximum for palette Dark2 is 8
## Returning the palette you asked for with that many colors

## Warning: Removed 3119 rows containing missing values (geom_point).
```

Kittitas	Lackawanna	Larimer	Leflore	Little River
Kittson	Laclede	Larue	Lehigh	Live Oak
Klamath	Lafayette	Las Animas	Lemhi	Livingston
Kleberg	Lafourche	Lassen	Lenawee	Llano
Klickitat	Lagrange	Latah	Lenoir	Logan
Knott	Lake	Latimer	Leon	Long
Knox	Lake and Peninsula	Lauderdale	Leslie	Lonoke
Kodiak Island	Lake of the Woods	Laurel	Letcher	Lorain
Koochiching	Lamar	Laurens	Levy	Los Alamos
Kootenai	Lamb	Lavaca	Lewis	Los Angeles
Kosciusko	Lamoille	Lawrence	Lewis and Clark	Loudon
Kossuth	Lampasas	Le Flore	Lexington	Loudoun County
La Crosse	Lancaster	Le Sueur	Lexington City	Louisa
La Moure	Lancaster County	Lea	Liberty	Louisa County
La Paz	Lander	Leake	Licking	Loup
La Plata	Lane	Leavenworth	Limestone	Love
La Porte	Langlade	Lebanon	Lincoln	Loving
La Salle	Lanier	Lee	Linn	Lowndes
Labette	Lapeer	Lee County	Lipscomb	Lubbock

corelation plot on single core

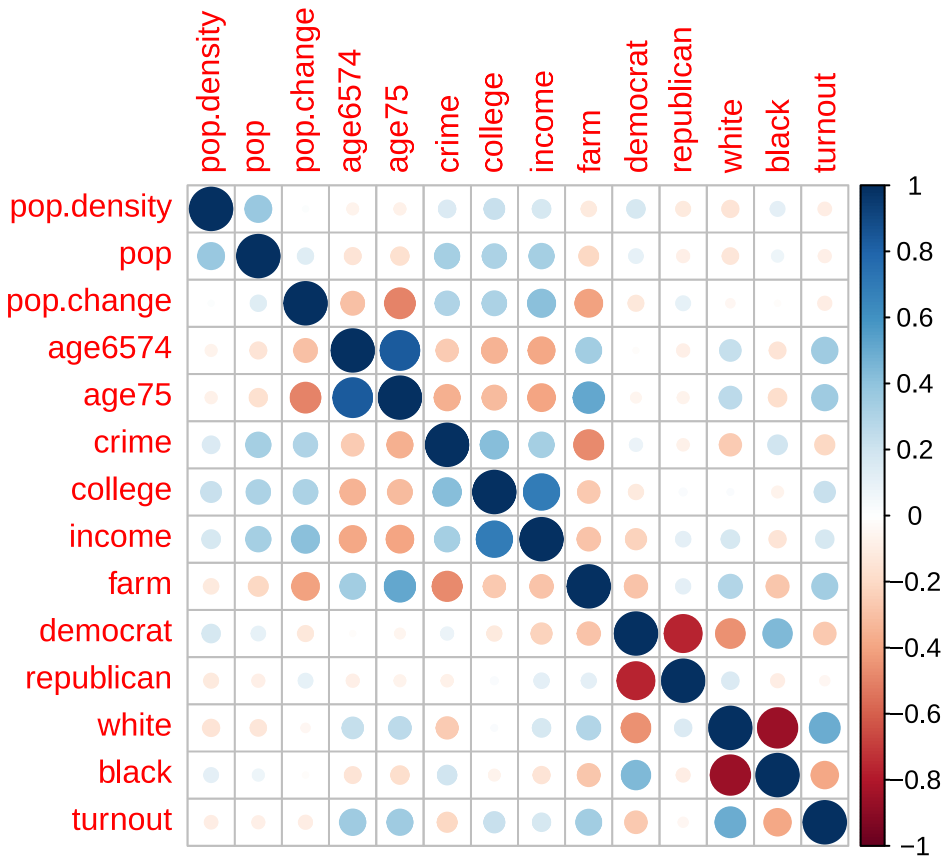
```
#install.packages("corrplot")

library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
df <- na.omit(df)
result = cor(df[,3:16])

system.time(corrplot(result, method = "circle"))
```



user system elapsed 0.008 0.000 0.008

2. Measuring time for mean computation on multiple cores For this purpose we can mcapply function from parallel package which is other version of lapply of base R. We will use 4 cores.

```
myfunc <- function(df) {

boxplot(df[,3],df[,4])

}
system.time(mclapply(list(df,df,df),myfunc, mc.cores=4,mc.set.seed=FALSE, mc.preschedule=FALSE) )
```

user system elapsed 0.003 0.009 0.021

```
myfunc <- function(df){

ggplot(data=df,aes(x=reorder(county,pop.density),y=income)) +
  geom_bar(stat ='identity',aes(fill=income))+
  coord_flip() +
  theme_grey() +
  scale_fill_gradient(name="Maths Score Level")+
  labs(title = 'income and population relation',
       y='population density',x='County')+
  geom_hline(yintercept = mean(df$income),size = 1, color = 'blue')


}
system.time( res <- mclapply(list(df,df,df),myfunc, mc.cores=4,mc.set.seed=FALSE, mc.preschedule=FALSE)
```

user system elapsed 0.024 0.023 0.086

```
myfunc <- function(df){
    df %>%
              group_by(income) %>%
              summarise()
}
```

using parallel version of lapply

```
set.seed(8)

system.time( res <- mclapply(list(df,df,df,df,df,df),myfunc, mc.cores=4,mc.set.seed=FALSE, mc.preschedul
```

user system elapsed 0.058 0.082 0.072

```
myfunc <- function(df){


result = cor(df[,3:16])
corrplot(result, method = "circle")

}
```

```
set.seed(1)
system.time(mclapply(list(df,df,df),myfunc, mc.cores=4,mc.set.seed=FALSE, mc.preschedule=FALSE) )
```

user system elapsed 0.033 0.044 0.050

We observe multi cores process takes less time as compared to single core