DATASET

Dota 2 - Pro Players Matches Results 2019 ~ 2021

Dota 2 is a multiplayer online battle arena (MOBA) video game in which two teams of five players compete to collectively destroy a large structure defended by the opposing team known as the "Ancient", whilst defending their own

- This dataset is long to be a given cross, once it passed through various stages of transformations, crosses and aggregations.
- The present information is statistics of each time one day before the departure in questão ter início.
- These statistics are calculated from the information of the games of each player not 6 months prior to the game in questão.
- Also, each line of this dataset has information on which time it gained at the start, as well as summary and 'non-normalized' statistics for each time.

The list of commands/script executed

```
➢ data<-read.csv("C:/Users/DELL/OneDrive/Desktop/eda.csv",header=TRUE,s
  ep=",")
```

```
title: "EDA LA2"
author: "ANKITA ASMA"
date: '2022-06-22'
output:
  html_document:
    df_print: paged
  word_document: default
  pdf_document: default
---


---
```

```
➢ The data in data frame can be read using read.csv()
```

```{r}
data<-read.csv(file.choose(),header=TRUE,sep=",")
```

```{r}
  data

```

- str() displays the internal structure of an r object

```{r}
str(data)
```

- summary() function used to produce result sumarries of the result of various model fitting functions

```{r}
summary(data)
```

- names() displays name of all the columns

```{r}
names(data)
```

- row.names() displays the row names in the dataframe

```{r}
row.names(data)
```

```
```

- rownames() displays the row names

```{r}
  rownames(data)
```

- colnames() displays the columns names in the dataframe

```{r}
   colnames(data)

```

- dimnames() displays both rows and columns names

```{r}
  dimnames(data)
```

- length()   gives the length of the selected attributed

```{r}
    length(data$match_id)
```

- max() gives the maximum value in the dataset

na.rm=True ignores all the NA values

```{r}
    max(data ,na.rm=TRUE)
```

- min() gives the minimum value in the dataset

```{r}
   min(data,na.rm=TRUE)
```

- sum() adds all the values in the dataset

```{r}
   sum(data,na.rm=TRUE)

```

- fivenum()-gives the quartile results without header

```{r}
  fivenum(data,na.rm=TRUE)
```

- quantile()-gives the quartile results with header

```{r}
  quantile(data,na.rm=TRUE)
```

- mean() gives mean of the column values

```{r}
   mean(data$deaths_avg_r,na.rm=TRUE)
```

- median() gives median of the column values

```{r}
    median(data$deaths_avg_r,na.rm=TRUE)
```

- table() gives the count of values in the dataset

```{r}
    table(data$ancient_kills_avg_r)
```

```{r}
   t(data)
```

- as.table() -converting dataframe to matrix and getting the count of the values

```{r}
   as.table(as.matrix(data))
```

- head() displays the first n records in the dataset

```{r}
head(data ,n=10)
```

- tail() displays the last n records in the dataset

```{r}
  tail(data,n=10)
```

- cumsum() -gives cumulative sum of the values in the dataset

```{r}
   cumsum(data)
```

```{r}
  cumsum(data$radiant_win)
```

- cummax()-gives cumulative maximum values in dataset

```{r}
  cummax(data$radiant_win)
```

- cummin()-gives cumulative minimum values in dataset

```{r}
   cummin(data$radiant_win)

```

- cumproduct()-gives cumulative maximum values in dataset

```{r}
  cumprod(data$radiant_win)
```

- sd()- it gives the std deviation of the values

```{r}
    sd(data$duration_avg_win_r,na.rm=TRUE)
```

- mad()- it gives median absolute deviation

```{r}
    mad(data,na.rm=TRUE)
```

- var() -gives the variance of the values

```{r}
     var(data,na.rm=TRUE)
```

- sort() sort the values of the columns in dataset

```{r}
     sort(data$match_id)
```

- order() sort in order and gives its index values

```{r}
    order(data$hero_kills_avg_r)
```

- rank() sort the values and gives its index value where it is originally present

```{r}
    rank(data$hero_kills_avg_r)
```

- stack() - to combine the vectors

```{r}
stack(data)
```

- unstack() - to segregate the stacked data

```{r}
  unstack(data)
```

- rowMeans() -gives the mean value of row

```{r}
      rowMeans(data)
```

```{r}
```

```
rowMeans(data ,na.rm=TRUE)

```
```

- colmeans() gives the mean value of columns

```
```{r}
colMeans(data ,na.rm=TRUE)
```
```

- rowsums() gives the sum of the values in row

```
```{r}
rowSums(data,na.rm=TRUE)
```
```

```
```{r}
matdata=as.matrix(data)
```
```

```
```{r}
matdata

```
```

- class() it displays which class it belongs

```
```{r}
class(matdata)
```
```
```

```{r}
str(matdata)
```

```{r}
matdata[3,3]
```

```{r}
matdata[3,1:4]
```

```{r}
matdata[,1]
```

```{r}
matdata[1,]
```

```{r}
matdata[c(1,3,5,7),]
```

```{r}
matdata[c(1,3,5,7),4]
```

```{r}
matdata[c(1,3,5,7),"hero_healing_avg_r"]
```

```
```

```r
matdata[3]
```

```r
sort(matdata)
```

```r
order(matdata[c(1,3,5,7),4])
```

```r
rank(matdata[c(1,3,5,7),4])
```

```r
sort(matdata[,1])

```

- as.list() -converting dataframe to list

```r
listdata=as.list(data)

```

```r
listdata
```

```r
str(listdata)
```

```{r}
class(listdata)
```

```{r}
listdata[1:4]
```

```{r}
listdata$match_id

```

```{r}
listdata$match_id[1:4]



```

```{r}
class(data)

```

```{r}
library(gcookbook)
library(ggplot2)
library(tidyr)
```

- loading the dplyr package

```{r}
library(dplyr)
```

```
```
%>% - pilpline operator

- mutate()-add new columns

```{r}
data %>% mutate(newcol=NA)
```

- as.character add new character

```{r}
char=as.character(data)

```
```

```{r}
char
```
```

```{r}
class(char)
```
```

- rep()-repeats the values

```{r}
vec<-rep(c(1,2,3),33)
data %>% mutate(newcol=vec)
```
```

```{r}
```

```
data2<-data

```

```{r}
data2
```

```{r}
data2$newcol<-c(1,2)
```

```{r}
data2$newcol

```

```{r}
data2$new<-(c(1,2,3),30)
```

- select()-selects a particular attribute - deletes the columns

```{r}
data %>% select(-firstblood_claimed_avg_r)
```

```{r}
data2 %>% select(-firstblood_claimed_avg_r,-newcol)
```

```
```

```{r}
data %>% select(hero_damage_avg_r)
```

- rename()-it renames the columns name

```{r}
sample <- data2 %>% rename(length=hero_damage_avg_r)
```

```{r}
sample
```

```{r}
names(data2)

```

```{r}
data2 %>% select(hero_damage_avg_r,hero_healing_avg_r,hero_kills_avg_r)

```

```{r}
data2 %>% select(hero_kills_avg_r,everything())
```

```{r}
data2[c("hero_damage_avg_r","hero_healing_avg_r")]
```

```{r}
data2[,"hero_kills_avg_r"]
```

```{r}
data2[, "hero_damage_avg_r", drop=FALSE]
```

- filter()-filters the data based on the given condition

```{r}
data2 %>% filter(radiant_win==TRUE & freq_r>=11 & freq_r<=100)
```

```{r}
data2 %>% filter(radiant_win==TRUE & freq_r>=11 & freq_r<=100) %>% select(match_id,radiant_win,freq_r)

```

```{r}
slice(data2,1:5)
```

```{r}
ggplot(data,aes(x=match_id,y=freq_r))+geom_point()
```

plot()-plots the graph

```{r}
plot(data$match_id,data$freq_id)

```

- ggplot() is an r package used for statistical computing and data representation using visualization

```{r}

ggplot(data,aes(x=match_id,y=freq_r))+geom_line()

```

- barplot()-plots the bar graph

```{r}
barplot(table(data$duration_avg_win_r))
```

```{r}
ggplot(data,aes(x=match_id,y=freq_r))+geom_col()

```

```{r}

plot(data$ancient_kills_avg_r, data$deaths_avg_r, type = "c")
```

```
```

```{r}

plot(data$ancient_kills_avg_r, data$deaths_avg_r, type = "p")
```

```{r}

plot(data$ancient_kills_avg_r, data$deaths_avg_r, type = "l")
```

```{r}
plot(data$radiant_win, data$freq_r, type = "l")
points(data$radiant_win, data$freq_r)
```

```{r}
plot(data$radiant_win, data$freq_r/2, col="red")
points(data$radiant_win, data$freq_r/2,col="red")
```

- geom_line()-plots the line graph
- geom_point()-plots the points in the graph

```{r}
ggplot(data, aes(x = win_pct_r, y = radiant_win)) +
 geom_line() +
 geom_point()

```

```{r}
ggplot(data, aes(x = factor(match_id), y = actions_per_min_avg_r)) +
 geom_col()

```



```{r}
ggplot(data, aes(x = freq_r)) +
 geom_bar()

```



```{r}
ggplot(data, aes(x = radiant_win)) +
 geom_bar()
```
```{r}
ggplot(data, aes(x = factor(win_pct_r))) +
 geom_bar()

```



```{r}
ggplot(data, aes(x = factor(radiant_win))) +
 geom_bar()
```



```{r}
hist(data$courier_kills_avg_r)

```

```{r}
ggplot(data, aes(x = match_id)) +
 geom_histogram()
```

```{r}
ggplot(data, aes(x = courier_kills_avg_r)) +
 geom_histogram()
```

```{r}
hist(data$win_pct_r)
```

- geom_histogram()- to view the distribution of one-dimensional data with a histogram.

```{r}
ggplot(data, aes(x = freq_r)) +
 geom_histogram()

```

```{r}
ggplot(data, aes(x = freq_r)) +
 geom_histogram(binwidth = 10)

```

```{r}
plot(data$match_id, data$buyback_count_avg_r)
```

```
```

- boxplot-t to create a box plot for comparing distributions.

```
```{r}
boxplot(data$match_id, data$buyback_count_avg_r)
```
```{r}
boxplot(data$duration_avg_lose_r, data$duration_avg_lose_r)
```
```{r}
boxplot(data$duration_avg_lose_r, data$duration_avg_lose_r,data$deaths_avg_r,data$firstblood_claimed_avg_r)
```
```{r}
boxplot(data$duration_avg_lose_r, data$duration_avg_lose_r,data$actions_per_min_avg_r)
```
```{r}
boxplot(duration_avg_win_r ~ duration_avg_lose_r, data = data)
```
```{r}
boxplot(duration_avg_win_r ~ duration_avg_lose_r+match_id, data = data)
```
```

```
```{r}
ggplot(data, aes(x = duration_avg_win_r, y = duration_avg_lose_r)) +
 geom_boxplot()

```
```{r}
ggplot(data, aes(x = interaction(duration_avg_win_r), y = duration_av
g_lose_r+match_id)) +
 geom_boxplot()

```
```

- curve() - to plot a function curve.

```
```{r}
curve(x^3 - 5*x, from = -10, to = 10)
```
```

OUTPUT INTERPRETATION

First we load our .csv dataset file, then we are performing various arithmetic commands such as

mean,median,standard deviation,min,max,sum,cummin,cummax,cumprod,cumsum

we have also done graph based command using ggplot,plot,lines,points etc