BUSINESS FORECASTING

VIDEO GAME SALES FORECASTING

Team:

Ayushi Mundhe
Karishma Sawant
Nakul Kuralkar
Sanket Gohel
Sinchana Mysore Eshwar
Sonal Pawar

1. Introduction

Across the globe, consoles have become famous for playing video games of different genres. Due to software and hardware advancements many different consoles are being developed. Every year, the video game industry churns out hundreds of titles and sells millions of units around the globe. Xbox (developed by Microsoft), Playstation (developed by Sony), Nintendo, and the personal computer are few of the platforms that have the biggest hit games from a variety of publishers.

In this report, the following questions of interest are addressed:

- Is there any evidence of differences between sales in different regions of the world?
- Are there any associations when comparing the genre of a game and the sales across different regions?
- Are there any associations when comparing different platforms and the sales across different regions?
- Does publishers have any impact on sales of video games?
- Forecast sales on highest selling genre.

A publisher must decide which genre of game he should sell in order to gain the maximum profit from the global market. The results in this paper may provide some insight to publishers that are trying to decide which type of console, genre, and region are best to sell the product. The results can also help to decide the marketing strategy for the game according to the game.

2. Background

VGChartz.com has data publicly available where total worldwide sales (in millions of units) are recorded with categorical information of genre, publisher, platform, and year of release for each title released. For all platforms that are no longer manufactured, sales are recorded as total units shipped from the manufacturers.
Records for over 16 billion units of sale from 1980 to 2017 are available in this sample and classified by genre, platform, and year of release.

Following are the variables in the dataset: -

- Rank - Ranking of overall sales (Unique Values - 1.7K)
- Name - The games name (Unique Values - 11493x)
- Platform - Platform of the games release

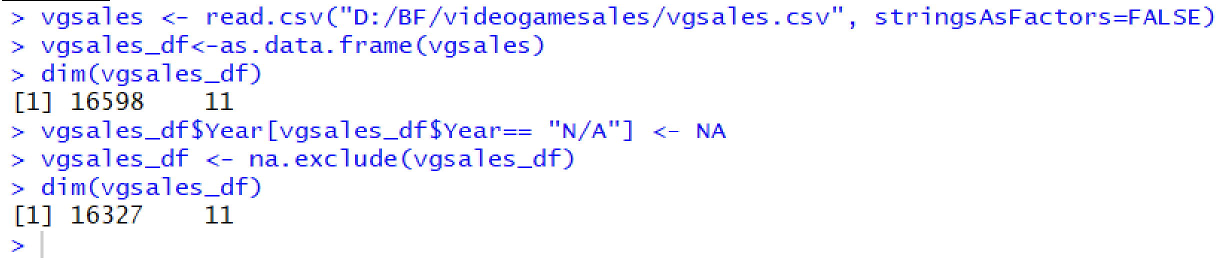
  (Unique Values - DS - 13%, PS2 - 13%, Other (29) - 74%)

- Year - Year of the game's release (Unique Values - 1.98k - 2.02k)
- Genre - Genre of the game (Unique Values - Action - 20%, Sports - 14%, Other - 66%)
- Publisher - Publisher of the game (Unique Values - Electronic Arts - 8%, Activision - 6%, Other (577) - 86%)
- NA_Sales - Sales in North America (Unique Values - 0 - 41.5, Values are in Millions.)
- EU_Sales - Sales in Europe (Unique Values - 0-29, Values are in millions)
- JP_Sales - Sales in Japan (Unique Values - 0 - 10.2, Values are in millions)
- Other_Sales - Sales in the rest of the world (Unique Values - 0 - 10.6, Values are in millions)
- Global_Sales - Total worldwide sales.(Unique Values - 0.01 - 82.7, Values are in Millions)

3. Data Cleaning

Before starting any analysis it is imperative to make sure that the data is in useable format and all the data entries are valid. As in the case of the Video Game data set, the column for year had some null values (N/A). The forecasting of our time series data is based on the yearly analysis of the sales in various regions. For this purpose, we have to clean the null values from the data set. Following is the code used for the same.

R Code:



```
> vgsales <- read.csv("D:/BF/videogamesales/vgsales.csv", stringsAsFactors=FALSE)
> vgsales_df<-as.data.frame(vgsales)
> dim(vgsales_df)
[1] 16598    11
> vgsales_df$Year[vgsales_df$Year== "N/A"] <- NA
> vgsales_df <- na.exclude(vgsales_df)
> dim(vgsales_df)
[1] 16327    11
> 
```

4. Data Exploration

Data Exploration is required to understand the dataset better. It can help choose appropriate variables in the dataset. The video game sales data has 11 variables. The sales seems to be varying in the regions depending on the genre of the game, publisher and the platform on which it has been released. In data exploration we'll try to determine which platforms are the best to sell video games, the best-selling genre and the publishers.

We will be using following libraries:

```
library(ggplot2)
library(magrittr)
library(dplyr)
require(reshape)
library(reshape2)
install.packages("dplyr", dep = TRUE)
```

4.1 Platform with highest Sales (Bar Graph):

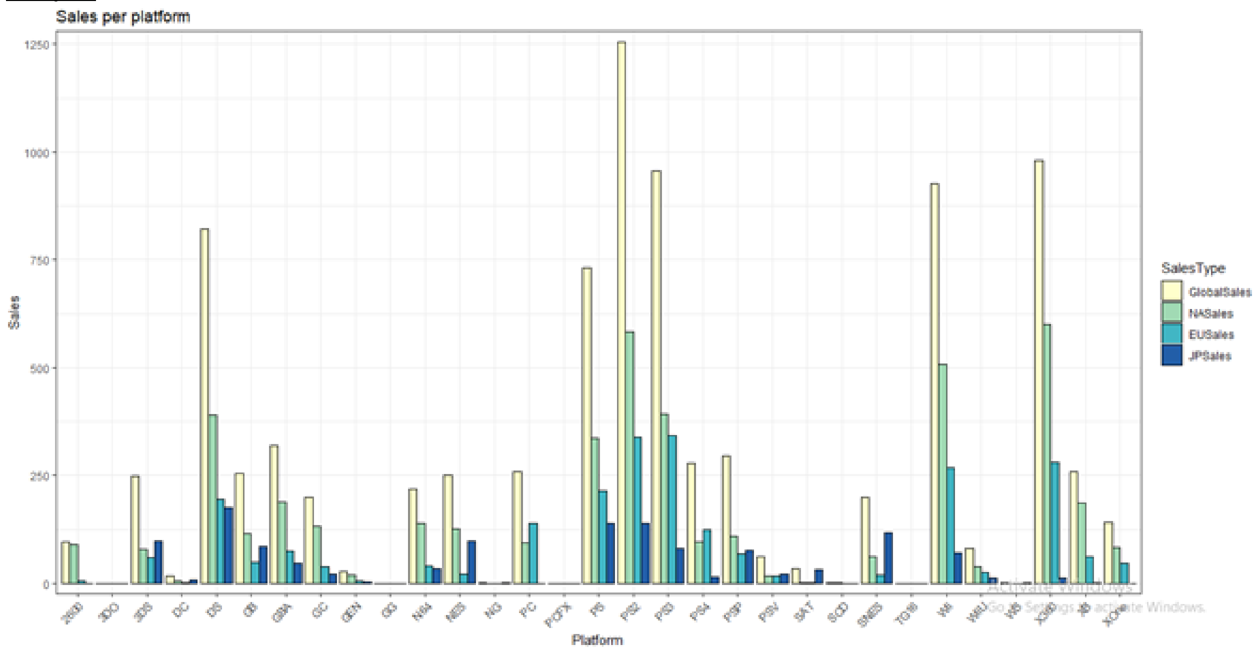
From the give data we are finding the platforms which have highest sales globally and in each region. The graph shows the following platforms having the highest sales:

DS, PS, PS2, PS3, Wii, X360

R Code:

```
vgsales <- read.csv("C:/Users/KURALKAR/Desktop/Business
Forecasting/videogamesales/vgsales.csv", stringsAsFactors=FALSE)
View(vgsales)  # reading the data from locally saved file
dim(vgsales)
vgsalesdf <- data.frame(vgsales)
names(vgsalesdf)

by_platform = group_by(vgsales, Platform)
sales_by_platform = melt(as.data.frame(sales_by_platform))
colnames(sales_by_platform) = c("Platform", "SalesType", "Sales")

p2 = ggplot(aes(x = Platform, y = Sales, fill = SalesType), data = sales_by_platform) +
geom_bar(colour = "black", stat = "identity", position = "dodge") +
theme_bw() + theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
ggtitle("Sales per platform") +
scale_fill_brewer(palette = "YlGnBu") + scale_y_continuous(expand = c(.02, .02)) + ylab("Sales")
P2
```

Graph:



4.2 Platform with highest Sales (Density Plot):

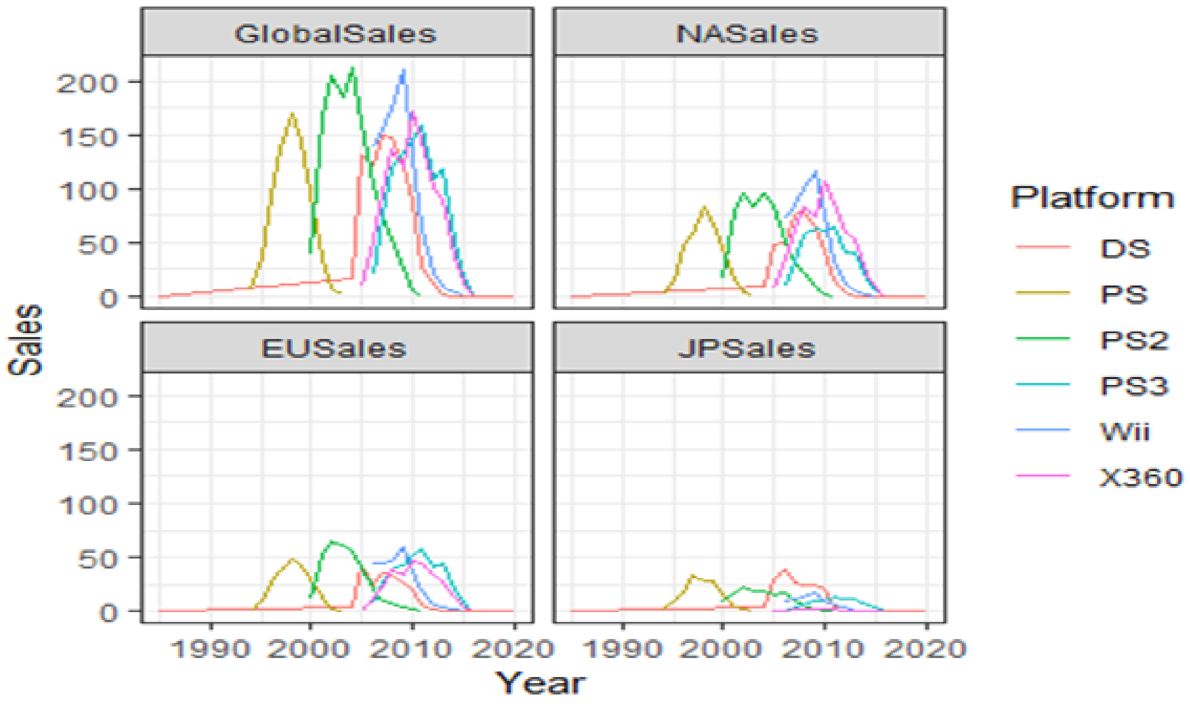
Taking the data from the highest selling platform, we can plot the sales line graph of for each region. As can be seen from the graph below, the GlobalSales and the NASales have similar patterns.

R Code:

```
sales_by_platform1 = vgsales %>%
    filter(Platform %in% c("DS", "PS", "PS2", "PS3", "Wii", "X360")) %>%
    group_by(Platform, Year) %>%
summarise(GlobalSales = sum(Global_Sales), NASales = sum(NA_Sales), EUSales =
sum(EU_Sales), JPSales = sum(JP_Sales))

sales_by_platform2 = melt(as.data.frame(sales_by_platform1))
colnames(sales_by_platform2) = c("Platform", "Year", "SalesType", "Sales")
sales_by_platform2$Year = as.numeric(sales_by_platform2$Year)
```

```
ggplot(aes(x = Year, y = Sales, colour = Platform, group = Platform), data = sales_by_platform2) +
 geom_line() + facet_wrap(~SalesType) + ylab("Sales") + theme_bw()
```

Graph:



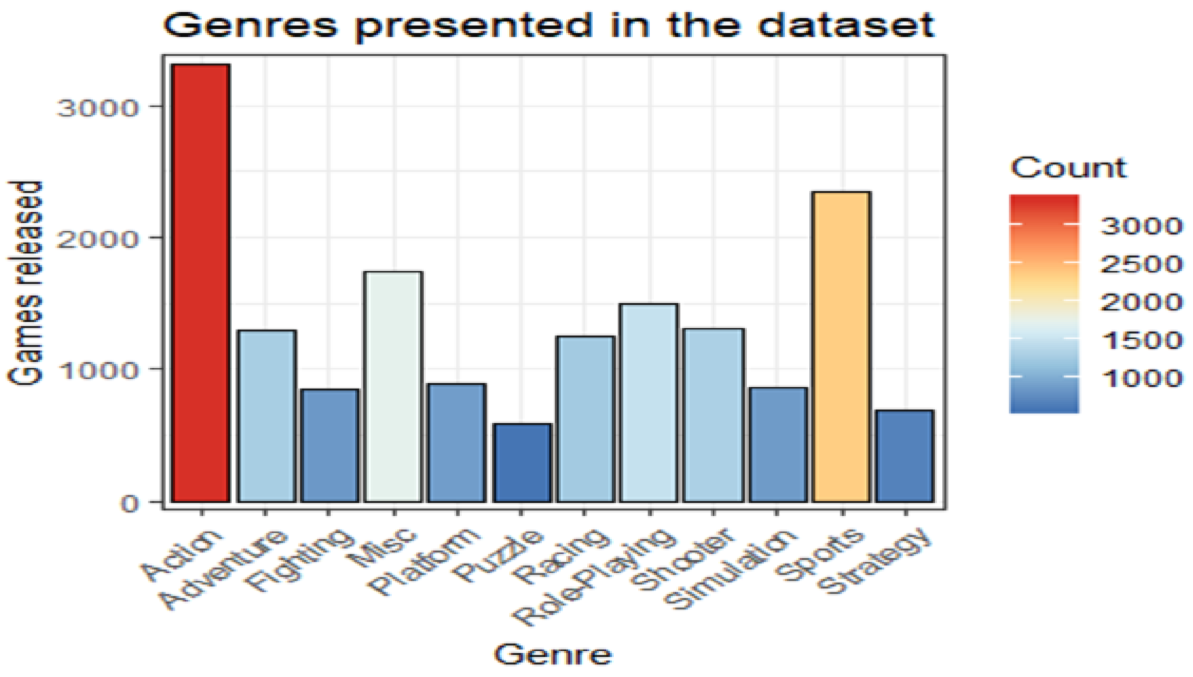
4.3 Highest selling genre:

The dataset has sales data of 11 types of genre. By grouping the dataset by genre, we can find the number of games released within each genre. As per the graph, the Action is the most preferred genre followed by Sports.

R Code:

```
by_genre = vgsales %>% group_by(Genre) %>% summarise(Count = n())
p3 = ggplot(aes(x = Genre, y = Count, fill = Count), data = by_genre) +
 geom_bar(, colour = "black", stat = "identity") +
 theme_bw() +
 theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
 ggtitle("Genres presented in the dataset") +
 scale_fill_distiller(palette = "RdYlBu") +
 scale_y_continuous(expand = c(.02, .02)) +
 ylab("Games released")
P3
```

Graph:



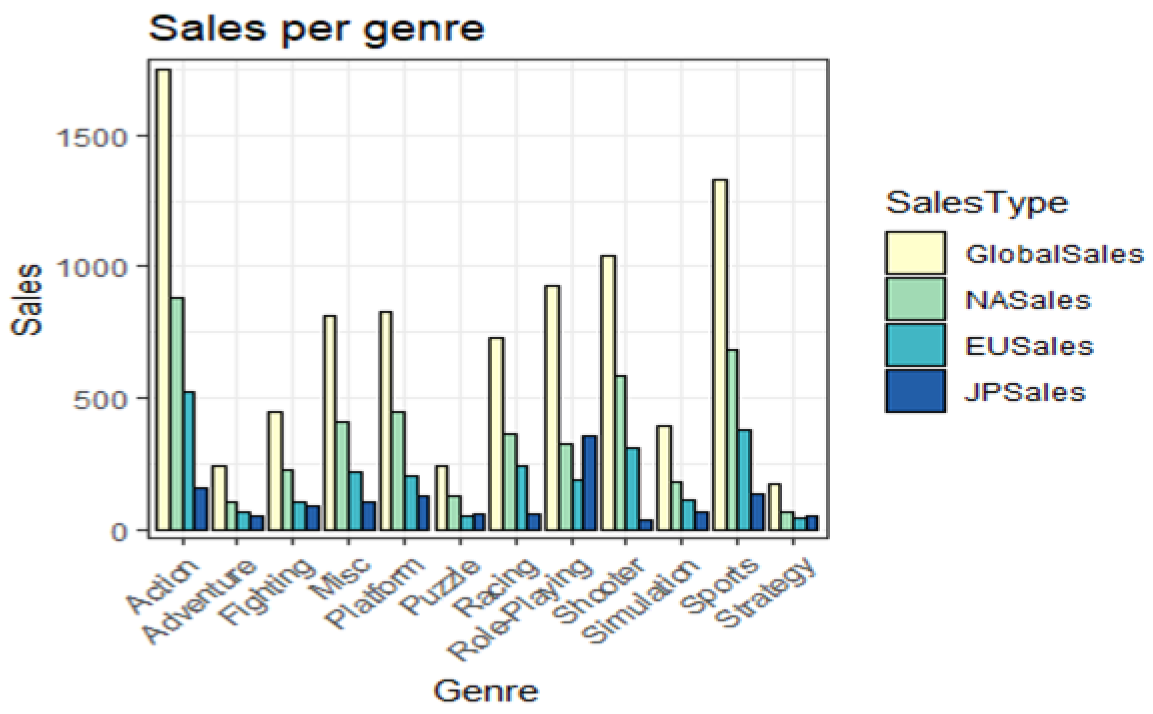
4.4 Sales per Genre:

By grouping the data by genre, the sales for each of the 11 genre can be found out. The graph below shows the sales for NA, EU and JP region along with the global sales for each genre. Again, the Action genre has the highest sales followed by Sports.

R Code:

```
sales_by_genre = vgsales %>% group_by(Genre) %>%
  summarise(GlobalSales = sum(Global_Sales), NASales = sum(NA_Sales), EUSales =
sum(EU_Sales),      JPSales = sum(JP_Sales))

sales_by_genre = melt(as.data.frame(sales_by_genre))
colnames(sales_by_genre) = c("Genre", "SalesType", "Sales")

p4 = ggplot(aes(x = Genre, y = Sales, fill = SalesType), data = sales_by_genre) +
  geom_bar(colour = "black", stat = "identity", position = "dodge") +
  theme_bw() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Sales per genre") +
  scale_fill_brewer(palette = "YlGnBu") +
  scale_y_continuous(expand = c(.02, .02)) +
  ylab("Sales")
P4
```

Graph:



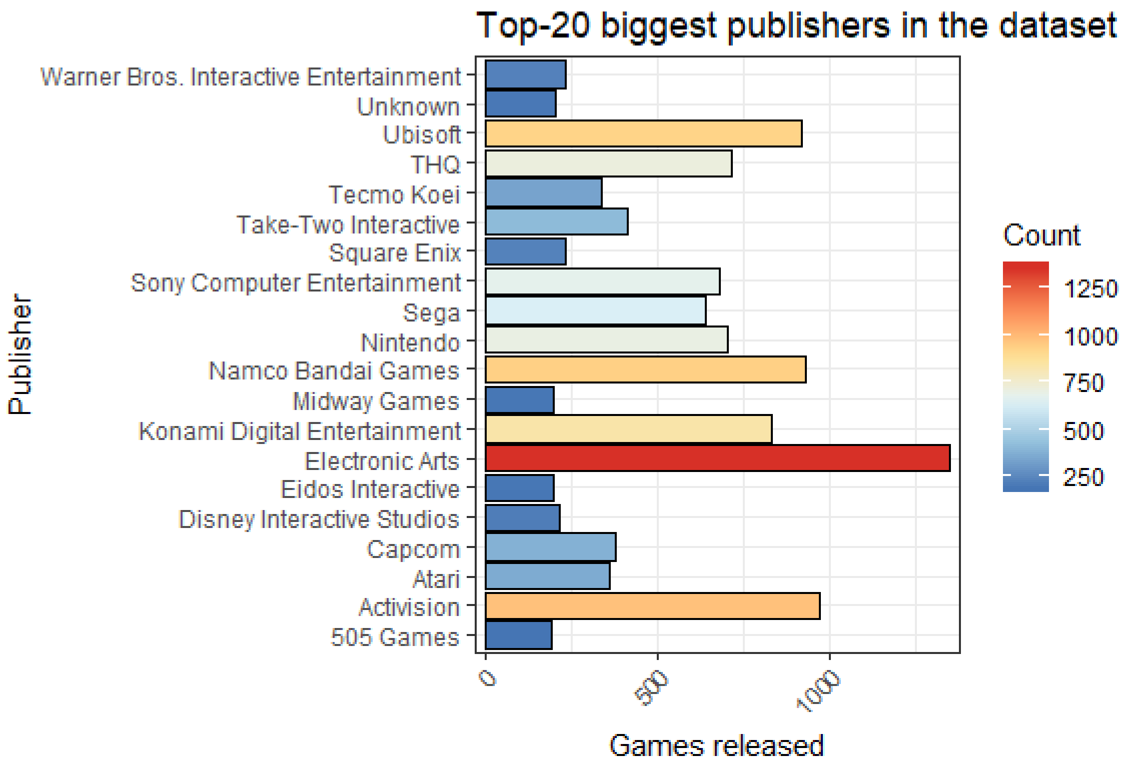
4.5 Top 20 biggest publishers:

The dataset has 578 unique publishers. Grouping the data by Publisher will lead us to find the top publishers who have released the most number of games. The following the graph shows the 20 biggest publishers with Electronics Arts being the top publisher.

R Code:

```
by_pub = vgsales %>% group_by(Publisher) %>% summarise(Count = n()) %>%
arrange(desc(Count)) %>% top_n(20)
p5 = ggplot(aes(x = Publisher, y = Count, fill = Count), data = by_pub) +
 geom_bar(, colour = "black", stat = "identity") + theme_bw() + theme(axis.text.x =
element_text(angle = 45, hjust = 1)) +
 ggtitle("Top-20 biggest publishers in the dataset") + scale_fill_distiller(palette = "RdYlBu") +
 scale_y_continuous(expand = c(.02, .02)) + ylab("Games released") + coord_flip()
p5
```

Graph:



After analyzing the data through EDA we took Action as the Genre and Year as a variable to forecast the data. Before forecasting and to fit a model we divided the dataset into Training and Testing Data.

When evaluating the quality of forecasts, it is invalid to look at how well a model fits the historical data; the accuracy of forecasts can only be determined by considering how well a model performs on new dataset that has not been used when fitting or training the model. When choosing models, it is common to use a portion of the available data for fitting and use the rest of the data for testing the model, as was done in the above examples.

5. Forecasting

5.1 Cyclic behavior

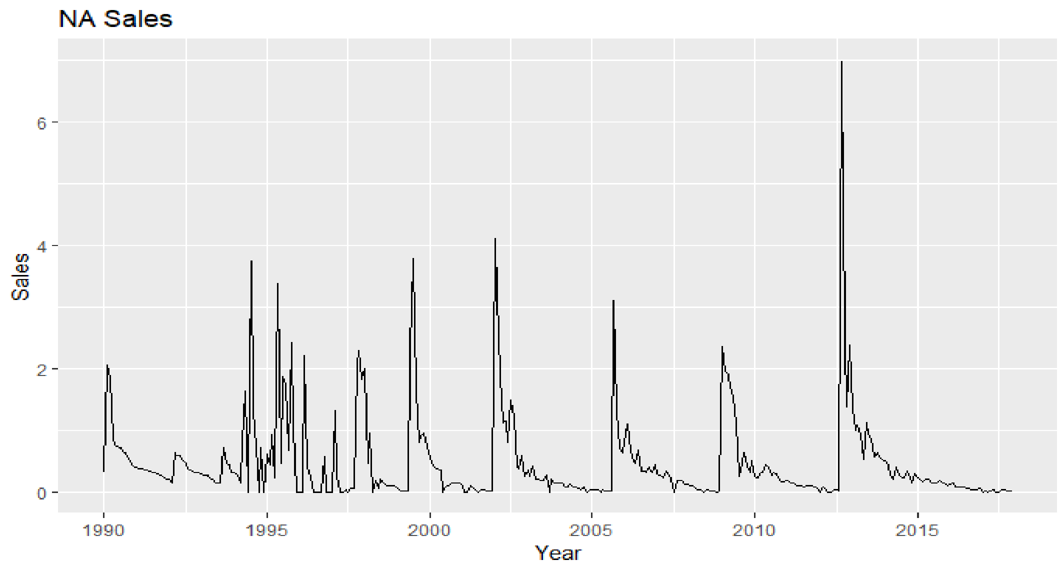
The cyclic behavior of data takes place when there are regular fluctuations in the data which usually last for an interval of at least two years, and when the length of the current cycle cannot be predetermined. Cyclic behavior is not to be confused with seasonal behavior. Seasonal fluctuations follow a consistent pattern each year so the period is always known. As an example, during the Christmas period, inventories of stores tend to increase in order to prepare for Christmas shoppers. As an example of cyclic behavior, the population of a particular natural ecosystem will exhibit cyclic behavior when the population increases as its natural food source decreases, and once the population is low, the food source will recover and the population will start to increase again. Cyclic data cannot be accounted for using ordinary seasonal adjustment since it is not of fixed period.

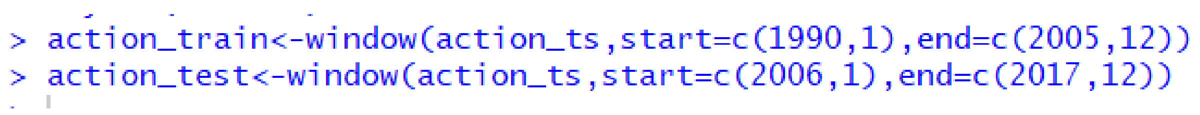
R Code:

```
> vgsales_ordered<-vgsales_df[order(as.Date(vgsales_df$Year, format="%Y")),]
> View(vgsales_ordered)
> vgsales_action<-vgsales_ordered[c(vgsales_df$Genre=='Action') ,  ]
> action_ts<-ts(vgsales_action,start=c(1990,1),end=c(2017,12),frequency = 12)
```

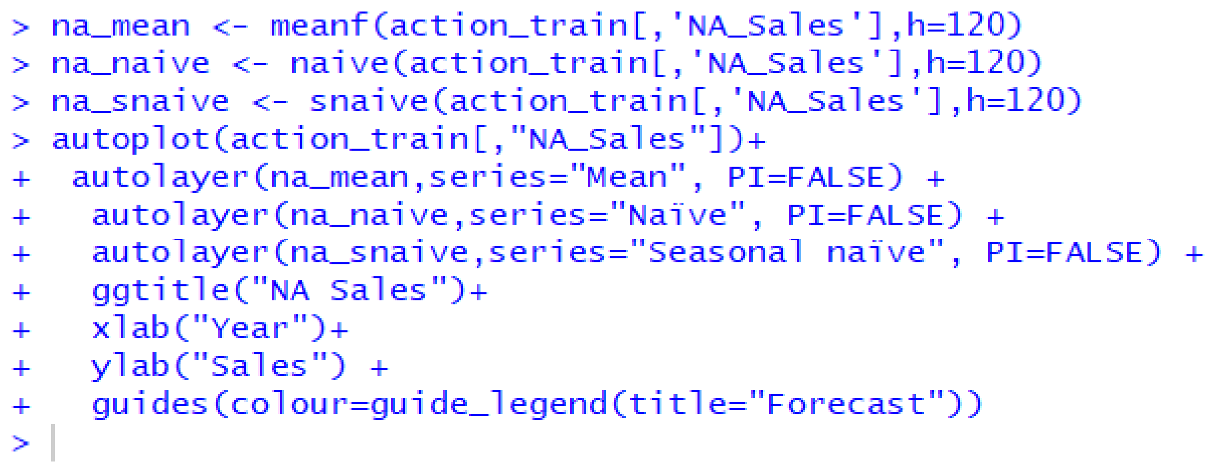
```
> autoplot(action_ts[,"NA_Sales"])+
+   ggtitle("NA Sales")+
+   xlab("Year")+
+   ylab("Sales")
```

Graph:





```
> action_train<-window(action_ts,start=c(1990,1),end=c(2005,12))
> action_test<-window(action_ts,start=c(2006,1),end=c(2017,12))
```



```
> na_mean <- meanf(action_train[,'NA_Sales'],h=120)
> na_naive <- naive(action_train[,'NA_Sales'],h=120)
> na_snaive <- snaive(action_train[,'NA_Sales'],h=120)
> autoplot(action_train[,"NA_Sales"])+
+  autolayer(na_mean,series="Mean", PI=FALSE) +
+   autolayer(na_naive,series="Naïve", PI=FALSE) +
+   autolayer(na_snaive,series="Seasonal naïve", PI=FALSE) +
+   ggtitle("NA Sales")+
+   xlab("Year")+
+   ylab("Sales") +
+   guides(colour=guide_legend(title="Forecast"))
> 
```

5.2 Mean, Naive and Seasonal Naive

Mean

In this approach, the predictions of all future values are equal to the mean of the past data. This approach can be used with any sort of data where past data is available. In time series notation:

$yT+h|T\}= y=(y1+...+yT)/T$

where $y1,...,yT$ iis the past data.

Although the time series notation has been used here, the average approach can also be used for cross-sectional data (when we are predicting unobserved values; values that are not included in the data set). Then, the prediction for unobserved values is the average of the observed values.

Naive

Naïve forecasts are the most cost-effective forecasting model, and provide a benchmark against which more sophisticated models can be compared. This forecasting method is only suitable for time series data Using the naïve approach, forecasts are produced that are equal to the last observed value. This method works quite well for economic and financial time series, which often have patterns that are difficult to reliably and accurately predict. If the time series is believed to have seasonality, the seasonal naïve approach may be more appropriate where the forecasts are equal to the value from last season. In time series notation:

$YT+h|T=yT$

Seasonal Naive

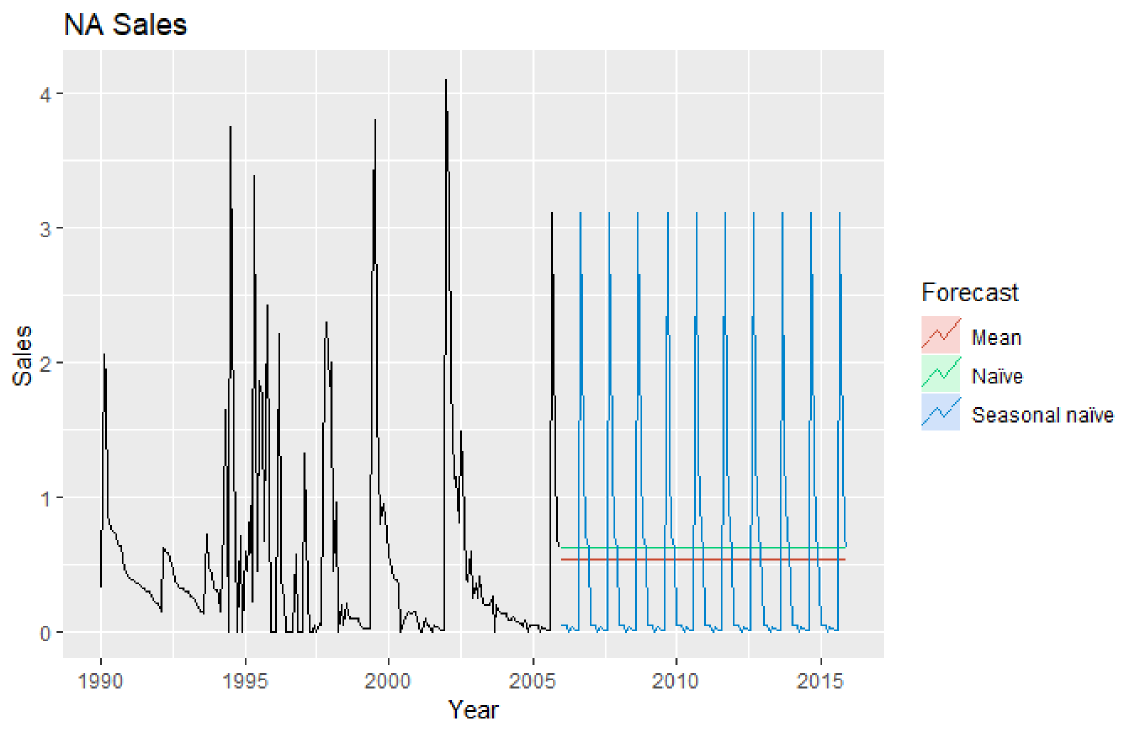
The seasonal naïve method accounts for seasonality by setting each prediction to be equal to the last observed value of the same season. For example, the prediction value for all subsequent months of April will be equal to the previous value observed for April. The forecast for time {\displaystyle T+h} is

YT+h|T=YT+h-km

Where, m=seasonal period and k is the smallest integer greater than (h-1)/m

The seasonal naïve method is particularly useful for data that has a very high level of seasonality.

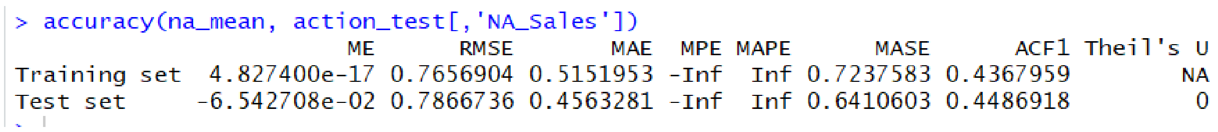
Graph:



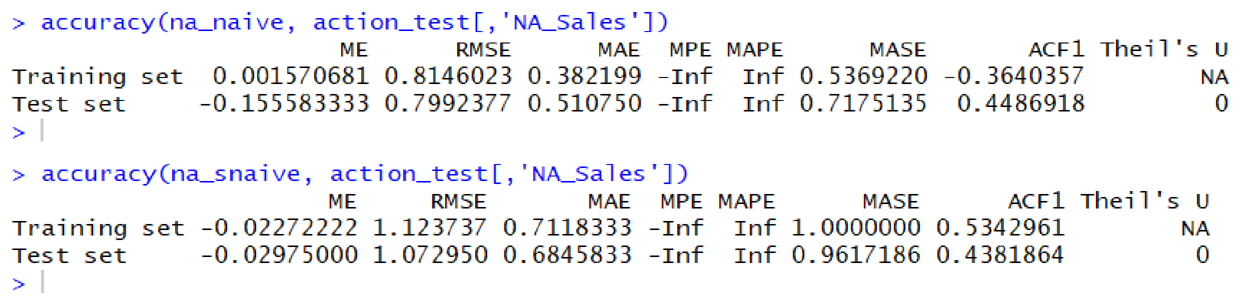
The above plot shows the mean, naive and seasonal naive forecasts for our training data set. The forecasts are done for the sales from the year 2006 till 2017.

- Naive forecast takes the last sales value for the year 2005
- Mean forecast is calculated by taking the mean of previous sales
- Seasonal Naive forecasts the seasonal trend of the year 2005

The RMSE values indicate that these models are certainly not the best fit for our data.



```
> accuracy(na_mean, action_test[,'NA_Sales'])
                         ME      RMSE       MAE  MPE MAPE      MASE      ACF1 Theil's U
Training set  4.827400e-17 0.7656904 0.5151953 -Inf  Inf 0.7237583 0.4367959        NA
Test set     -6.542708e-02 0.7866736 0.4563281 -Inf  Inf 0.6410603 0.4486918         0
```


```
> accuracy(na_naive, action_test[,'NA_Sales'])
                       ME      RMSE      MAE  MPE MAPE      MASE       ACF1 Theil's U
Training set  0.001570681 0.8146023 0.382199 -Inf  Inf 0.5369220 -0.3640357        NA
Test set     -0.155583333 0.7992377 0.510750 -Inf  Inf 0.7175135  0.4486918         0
> 
> accuracy(na_snaive, action_test[,'NA_Sales'])
                      ME     RMSE       MAE  MPE MAPE      MASE      ACF1 Theil's U
Training set -0.02272222 1.123737 0.7118333 -Inf  Inf 1.0000000 0.5342961        NA
Test set     -0.02975000 1.072950 0.6845833 -Inf  Inf 0.9617186 0.4381864         0
> 
```

5.3 STL Decomposition:

Time series decomposition is a mathematical procedure which transforms a time series into multiple different time series. The original time series is often split into 3 component series:

Seasonal: Patterns that repeat with a fixed period of time. For example, a website might receive more visits during weekends; this would produce data with a seasonality of 7 days.

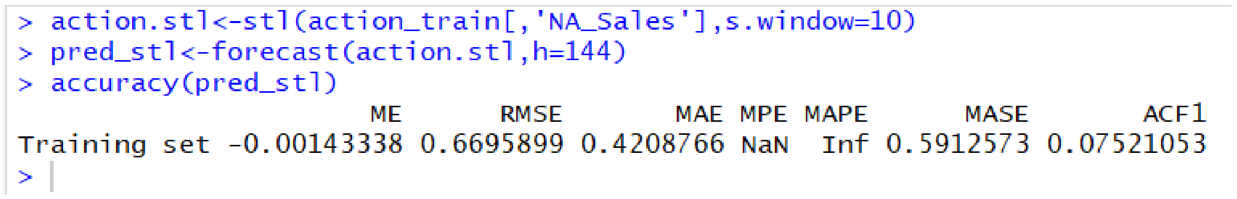
Trend: The underlying trend of the metrics. A website increasing in popularity should show a general trend that goes up.

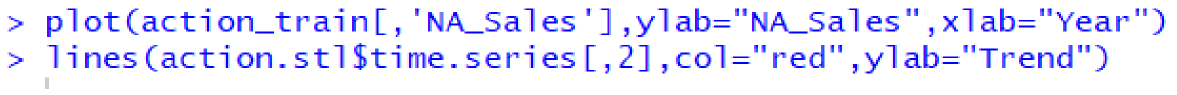
Random: Also call “noise”, “irregular” or “remainder,” this is the residuals of the original time series after the seasonal and trend series are removed.

According to the following graph, shows the NA sales data for the Action Genre.

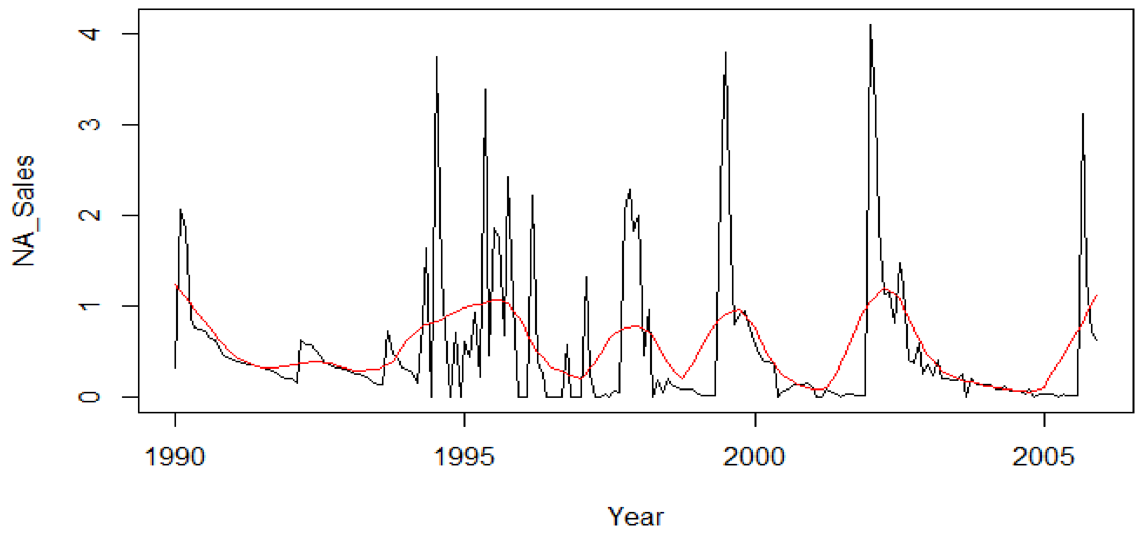
After applying STL on the training data, we observe no trend or seasonal pattern in the NA sales. The cyclic pattern in the data is shown by the red graph overlapping the original plot.

R Code:


```
> action.stl<-stl(action_train[,'NA_Sales'],s.window=10)
> pred_stl<-forecast(action.stl,h=144)
> accuracy(pred_stl)
                      ME      RMSE       MAE MPE MAPE      MASE       ACF1
Training set -0.00143338 0.6695899 0.4208766 NaN  Inf 0.5912573 0.07521053
> 
```


```
> plot(action_train[,'NA_Sales'],ylab="NA_Sales",xlab="Year")
> lines(action.stl$time.series[,2],col="red",ylab="Trend")
```

Graph:



5.4 Simple Exponential Smoothing

The simplest of exponential smoothing methods is simple exponential smoothing (SES). This method is suitable for forecasting data with no clear trend or seasonal pattern.

Using the average method, all future forecasts are equal to a simple average of the observed data,

$$\hat{y}_{T+h|T} = \frac{1}{T}\sum_{t=1}^{T} y_t,$$

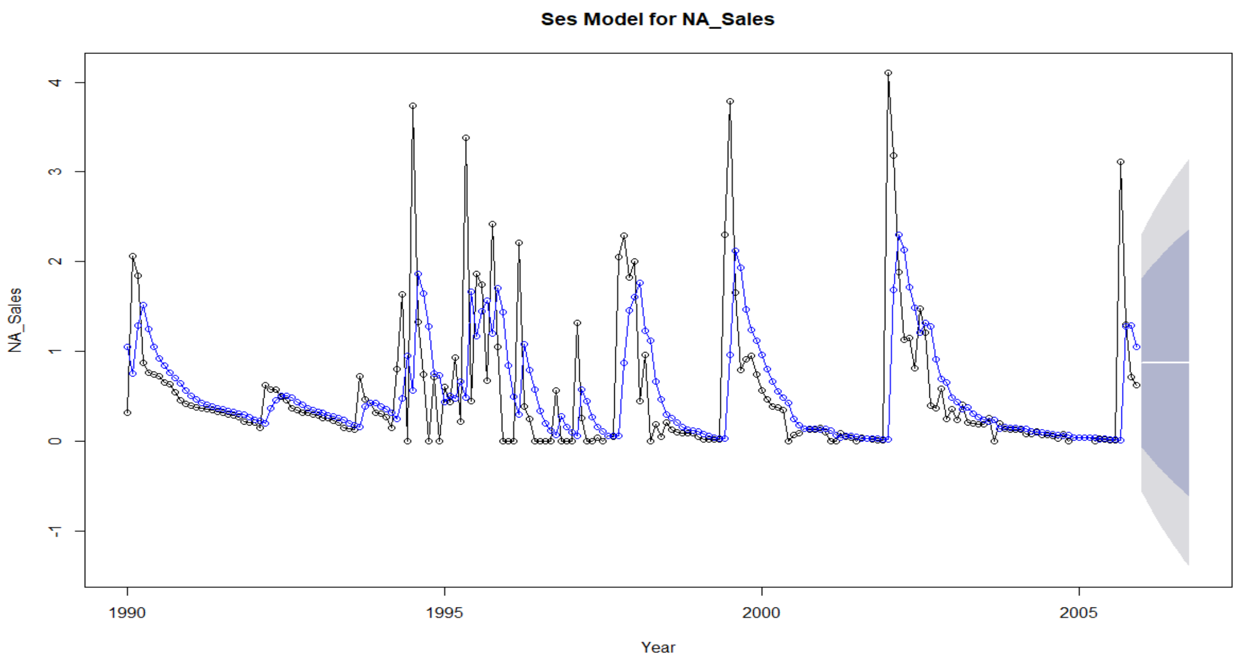
for h=1,2,…h=1,2,…. Hence, the average method assumes that all observations are of equal importance, and gives them equal weights when generating forecasts.

R Code:

```
> action.ses<-ses(action_train[,'NA_Sales'])
> action.ses$model
Simple exponential smoothing

Call:
 ses(y = action_train[, "NA_Sales"])

  Smoothing parameters:
    alpha = 0.4095

  Initial states:
    l = 1.0517

  sigma:  0.7332

     AIC     AICc      BIC
894.2568 894.3845 904.0293
```

```
> accuracy(action_train[,'NA_Sales'],fitted(action.ses))
                  ME      RMSE       MAE       MPE     MAPE      ACF1 Theil's U
Test set 0.002243953 0.7293665 0.3780001 -307.7615 375.3344 0.0862912  2.208019
> pred.ses<-forecast(action.ses,n.ahead = 12*12)
> accuracy(pred.ses)
                      ME      RMSE       MAE  MPE MAPE      MASE      ACF1
Training set -0.002243953 0.7293665 0.3780001 -Inf  Inf 0.5310233 0.0862912
```

```
> plot(action.ses, main="Ses Model for NA_Sales", ylab="NA_Sales", xlab="Year", fcol="white", type="o")
> lines(fitted(action.ses), col="blue", type="o")
```



As there was no Trend and Seasonality in our data, we used simple Exponential smoothing technique to create the model. After creating the model, we got the alpha value as 0.49, which should be between 0 to 1. Furthermore, we focus on AIC which is 894.25. As the AIC value in SES is too high, we create another model called ARIMA model.

5.5 ARIMA Model

While exponential smoothing models are based on a description of the trend and seasonality in the data, ARIMA models aim to describe the autocorrelations in the data.

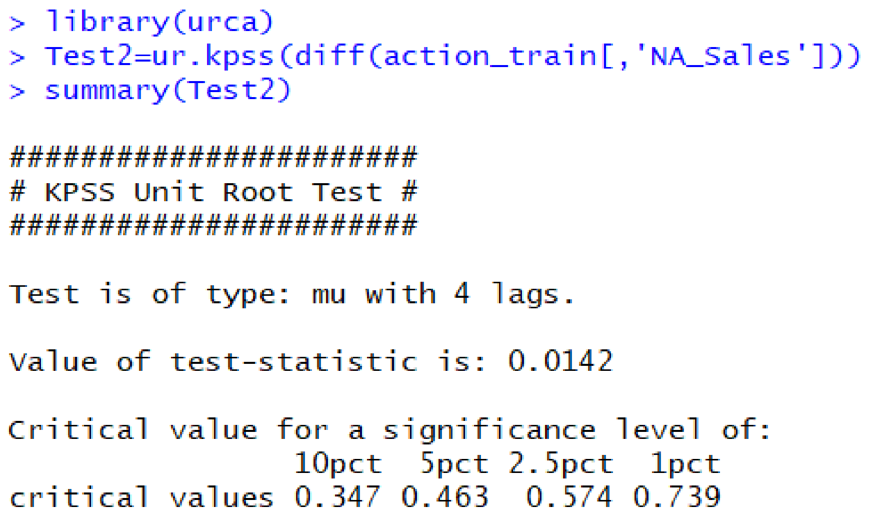
Kpss Test

The Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test figures out if a time series is stationary around a mean or linear trend or is non-stationary due to a unit root. A stationary time series is one where statistical property — like the mean and variance — are constant over time.

The null hypothesis for the test is that the data is stationary.

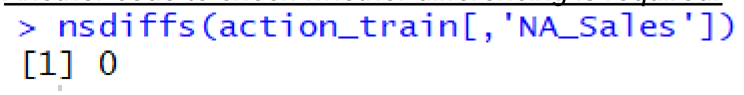
The alternate hypothesis for the test is that the data is not stationary.

R Code:



```
> library(urca)
> Test2=ur.kpss(diff(action_train[,'NA_Sales']))
> summary(Test2)

#######################
# KPSS Unit Root Test #
#######################

Test is of type: mu with 4 lags.

Value of test-statistic is: 0.0142

Critical value for a significance level of:
                10pct  5pct 2.5pct  1pct
critical values 0.347 0.463  0.574 0.739
```

Another code to check whether differencing is required:



```
> nsdiffs(action_train[,'NA_Sales'])
[1] 0
```

Autoregressive Models

In a multiple regression model, we forecast the variable of interest using a linear combination of predictors. In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable. The term autoregression indicates that it is regression of the variable against itself.

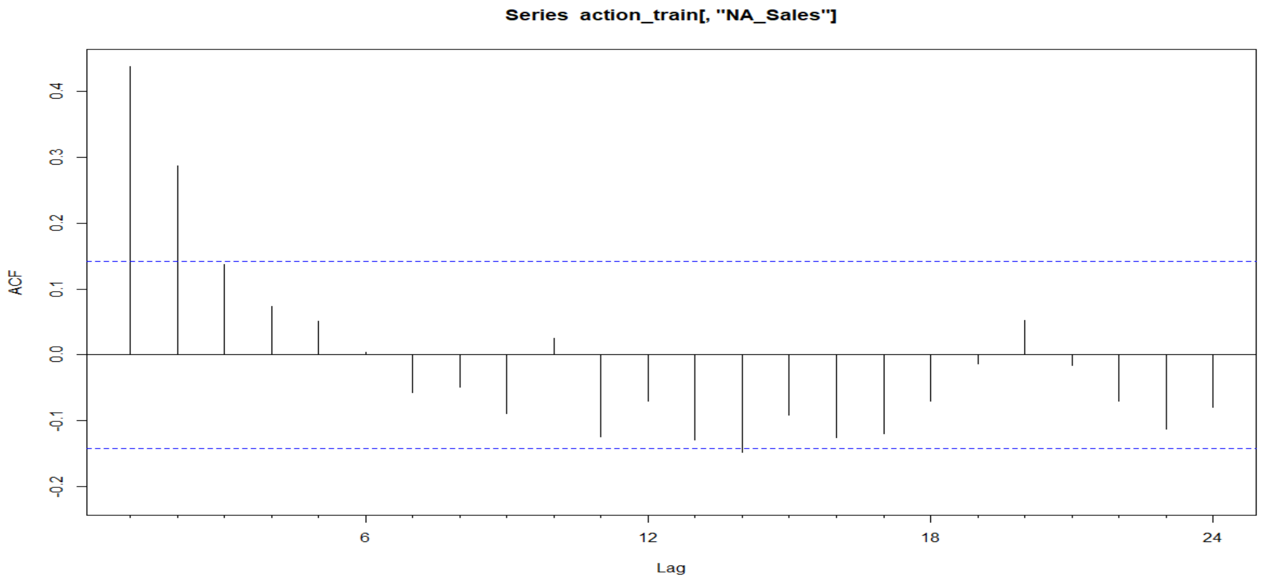
Thus, an autoregressive model of order p can be written as

$$y_t = c + \phi_1 y_{t-1} + \phi_2 y_{t-2} + \cdots + \phi_p y_{t-p} + \varepsilon_t,$$

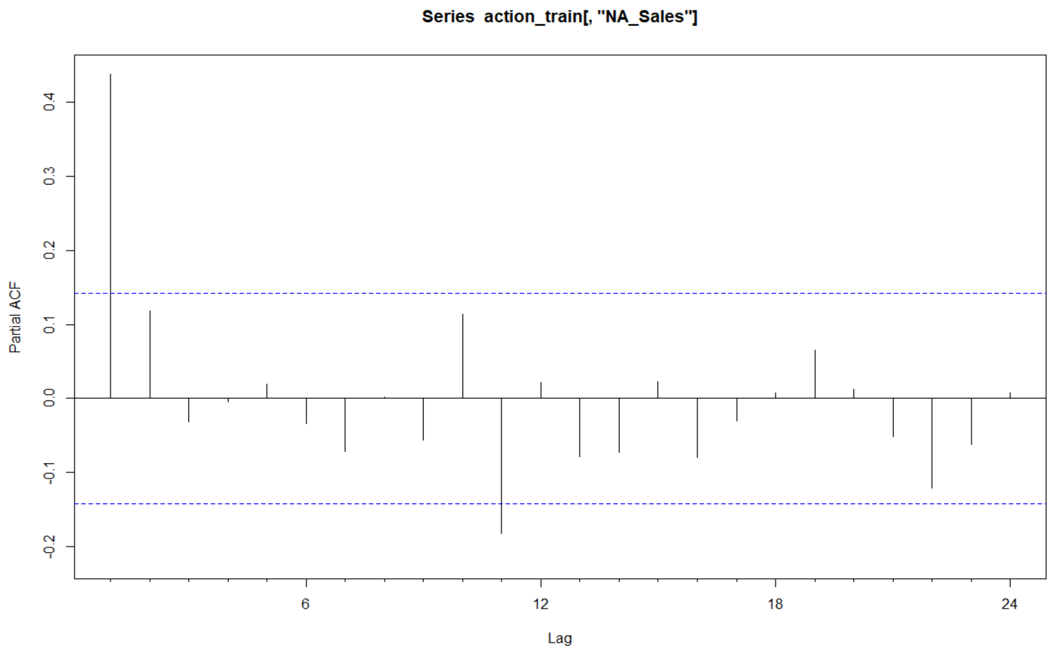
where εt is white noise. This is like a multiple regression but with *lagged values* of yt as predictors. We refer to this as an **AR(**p**) model**, an autoregressive model of order p.

Applying Acf and Pacf function to determine the lagged values-

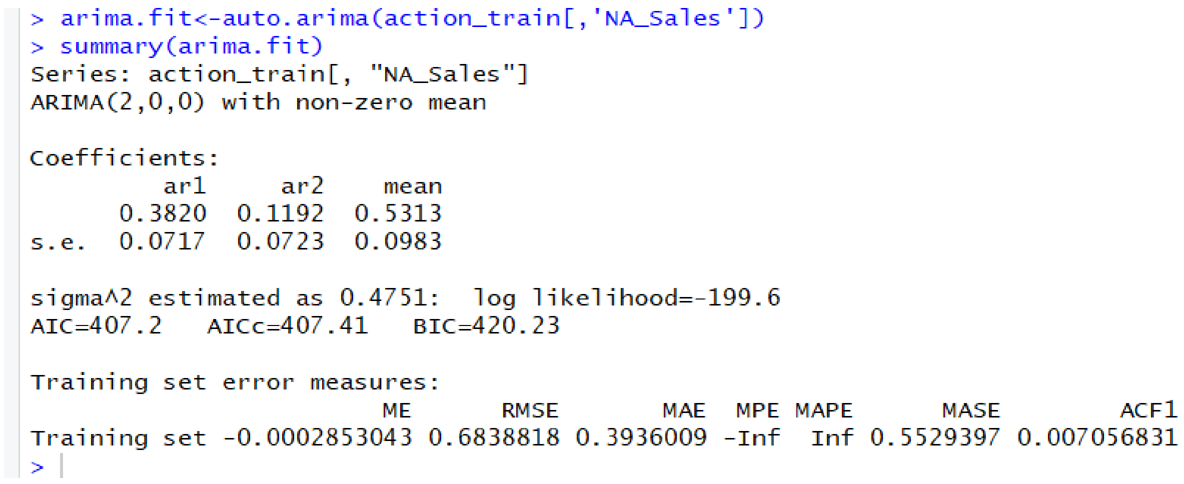
> Acf(action_train[,'NA_Sales'])



> Pacf(action_train[,'NA_Sales'])



Applying the auto.arima() on the training data gives the following results.

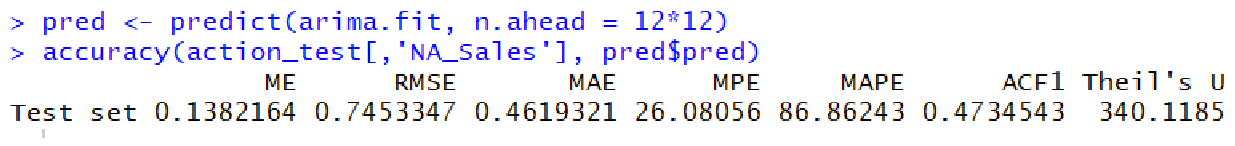


```
> arima.fit<-auto.arima(action_train[,'NA_Sales'])
> summary(arima.fit)
Series: action_train[, "NA_Sales"]
ARIMA(2,0,0) with non-zero mean

Coefficients:
         ar1     ar2    mean
      0.3820  0.1192  0.5313
s.e.  0.0717  0.0723  0.0983

sigma^2 estimated as 0.4751:  log likelihood=-199.6
AIC=407.2   AICc=407.41   BIC=420.23

Training set error measures:
                        ME      RMSE       MAE  MPE MAPE      MASE        ACF1
Training set -0.0002853043 0.6838818 0.3936009 -Inf  Inf 0.5529397 0.007056831
> 
```

auto.arima() suggests a P value of 2 with the d and q values equal to 0. This means that the differencing and moving average will not be applied to the dataset.

Using the values in the output, the autoregressive model can be written as:

$$y_t = C + 0.382 \times y_{t-1} + 0.1192 \times y_{t-2} + \varepsilon_t$$



```
> pred <- predict(arima.fit, n.ahead = 12*12)
> accuracy(action_test[,'NA_Sales'], pred$pred)
                ME      RMSE       MAE      MPE     MAPE      ACF1 Theil's U
Test set 0.1382164 0.7453347 0.4619321 26.08056 86.86243 0.4734543  340.1185
```

6. Conclusion

- The exploratory data analysis of the given data set gave us the best-selling genre i.e Action for all the regions. Also, grouping the dataset gave the best performing platforms and publishers.

- The data forecasting using Naive, Mean, Seasonal mean, SES and ARIMA models gave the sales forecast for the NA region for Action genre.

- The output for SES shows that it has an AIC value of 897. The non-seasonal ARIMA model has an AIC value of 407.2 which is much less than the SES model.

- The smaller the AIC value the better the model. Since ARIMA has a lower AIC value, it fits our database better than SES**.**