Final Project: Analysis of Video Games Sales

Hai Le and Zack Frazier

5/8/2020

Welcome to our tutorial! Are you a video games lover/enthusiast, a game developer with the aspiration of making millions, or just have a quest for knowledge for the business aspect of video games? Given the appropriate data, analysis via the power of **Data Science** can provide a deep insights into almost any real-life applications, including video games! Questions like how the sales of a specific genre fluctuated over time; what is the most popular video game publisher; what factor constitutes a high grossing video games, and many others can be answer with important data science concepts!

Sounds exciting right? Now let's talk **Data Science**!!

Data Science consists of a set of tasks that convert raw data into human readable information. The data science pipeline consists of: data curation, parsing, and management; exploratory data analysis; hypothesis testing and machine learning to provide analysis. In this tutorial, we will walk you through these 5 important steps to construct a successful project!

Here are the steps:

1. **Find** and **collect** data
2. **Proccess** the data, **tidy** the data, and deal with missing data
3. **Exploratory analysis** and **data visualization**
4. Perform **analysis**, **hypothesis testing**, and **machine learning**
5. Curation of a message or messages covering **insights** learned via the 4 steps above.

Step 1: Find and Collect the Data

Finding relevant data to your probelm in the first implied step! In this case, we are diving in to the business end of video games.

Data set URL: https://www.kaggle.com/gregorut/videogamesales/data

The website that we used: kaggle.come is a great source of data for a wide range of topics! The dataset that we chose contains a list of video games with sales greater than 100,000 copies. The different fields included within this dataset are:

- Rank - Ranking of overall sales
- Name - The games name
- Platform - Platform of the games release (i.e. PC, PS4, etc.)
- Year - Year of the game's release
- Genre - Genre of the game
- Publisher - Publisher of the game
- NA_Sales - Sales in North America (in millions)
- EU_Sales - Sales in Europe (in millions)
- JP_Sales - Sales in Japan (in millions)
- Other_Sales - Sales in the rest of the world (in millions)
- Global_Sales - Total worldwide sales.

How will we collect this set of data?

This project will be in R; therefore you will need to go to the installation proccess for R and Rstudio. After you've installed R and Rstudio, you will need the following libraries for the tasks that we will be covering:

- tidyverse
- rvest
- ggplot2
- lubridate
- caret
- e1071
- party

These libraries can be installed via this command: install.packages("library_name") in your RStudio's console. Replace the library_name with the libraries metioned above.

Now let us load in the raw data in RStudio!

```
raw_data <- read.csv("C:/Users/letha/Desktop/CMSC320/vgsales.csv")
head(raw_data)
```

```
##   Rank                     Name Platform Year        Genre Publisher NA_Sales
## 1    1               Wii Sports      Wii 2006       Sports  Nintendo    41.49
## 2    2        Super Mario Bros.      NES 1985     Platform  Nintendo    29.08
## 3    3           Mario Kart Wii      Wii 2008       Racing  Nintendo    15.85
## 4    4        Wii Sports Resort      Wii 2009       Sports  Nintendo    15.75
## 5    5 Pokemon Red/Pokemon Blue       GB 1996 Role-Playing  Nintendo    11.27
## 6    6                   Tetris       GB 1989       Puzzle  Nintendo    23.20
##   EU_Sales JP_Sales Other_Sales Global_Sales
## 1    29.02     3.77        8.46        82.74
## 2     3.58     6.81        0.77        40.24
## 3    12.88     3.79        3.31        35.82
## 4    11.01     3.28        2.96        33.00
## 5     8.89    10.22        1.00        31.37
## 6     2.26     4.22        0.58        30.26
```

Step 2: Data Processing

Looking at the raw data above, there are some N/A entries and unwanted columns. We will properly encode the N/A entries and remove the unwanted columns ~ Rank, NA_Sales, EU_Sales, JP_Sales, Other_sales

```
tidy_data <- raw_data
# cleaning the N/A entries
tidy_data[tidy_data == "N/A"] = NA
tidy_data <- drop_na(tidy_data)

# removing unwanted columns
tidy_data <- tidy_data[-c(1,7,8,9,10)]
head(tidy_data)
```

```
##                       Name Platform Year        Genre Publisher Global_Sales
## 1               Wii Sports      Wii 2006       Sports  Nintendo        82.74
## 2        Super Mario Bros.      NES 1985     Platform  Nintendo        40.24
## 3           Mario Kart Wii      Wii 2008       Racing  Nintendo        35.82
## 4        Wii Sports Resort      Wii 2009       Sports  Nintendo        33.00
## 5 Pokemon Red/Pokemon Blue       GB 1996 Role-Playing  Nintendo        31.37
## 6                   Tetris       GB 1989       Puzzle  Nintendo        30.26
```

Step 3: Exploratory Analysis and Data Visualization

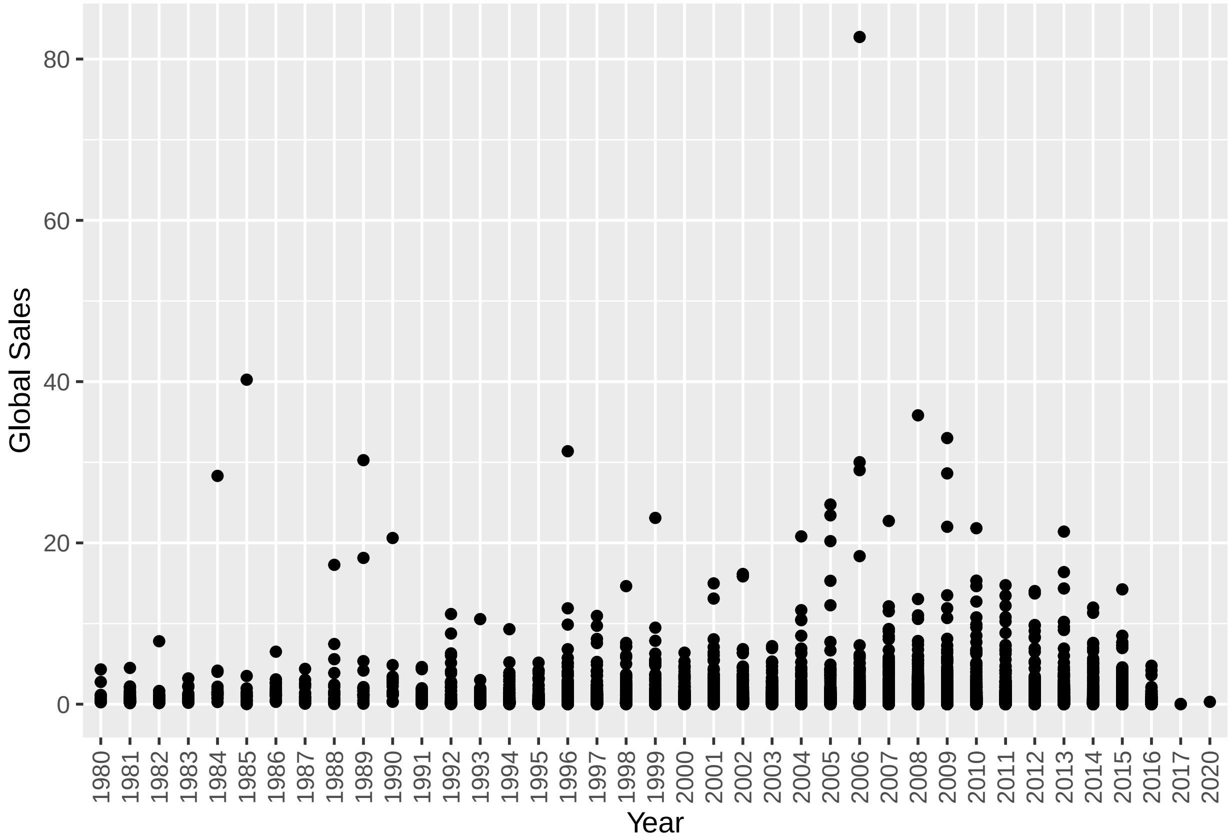
Exploratory Data Analysis is the last step before statistical analysis and machine learning. In this steps, we will cover the basics of visualizing datas through different graphing techniques and applications of linear regressions. Analyzing how pairs of variables interact gives important insights into different aspects of your data. Furthermore, data analysis will help us to find assumptions for the predications we will be making.

We are going to visualize the raw data in a variety of different *scopes*, including:
* Global Sales vs. Years * Global Sales vs. Different Genres * Global Sales vs. Different Publishers

Global Sales vs. Years

First, let's take a look at the relationship between global sales and years. Has the video game businesses gathered more money as time goes by? or has it decreased? This question can be answered after we analyze/visualize global sales vs. year.

```r
plot <- tidy_data %>%
  ggplot(mapping = aes(x= paste(Year), y = Global_Sales)) +
  geom_point()
plot +
  xlab("Year") + ylab("Global Sales") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```



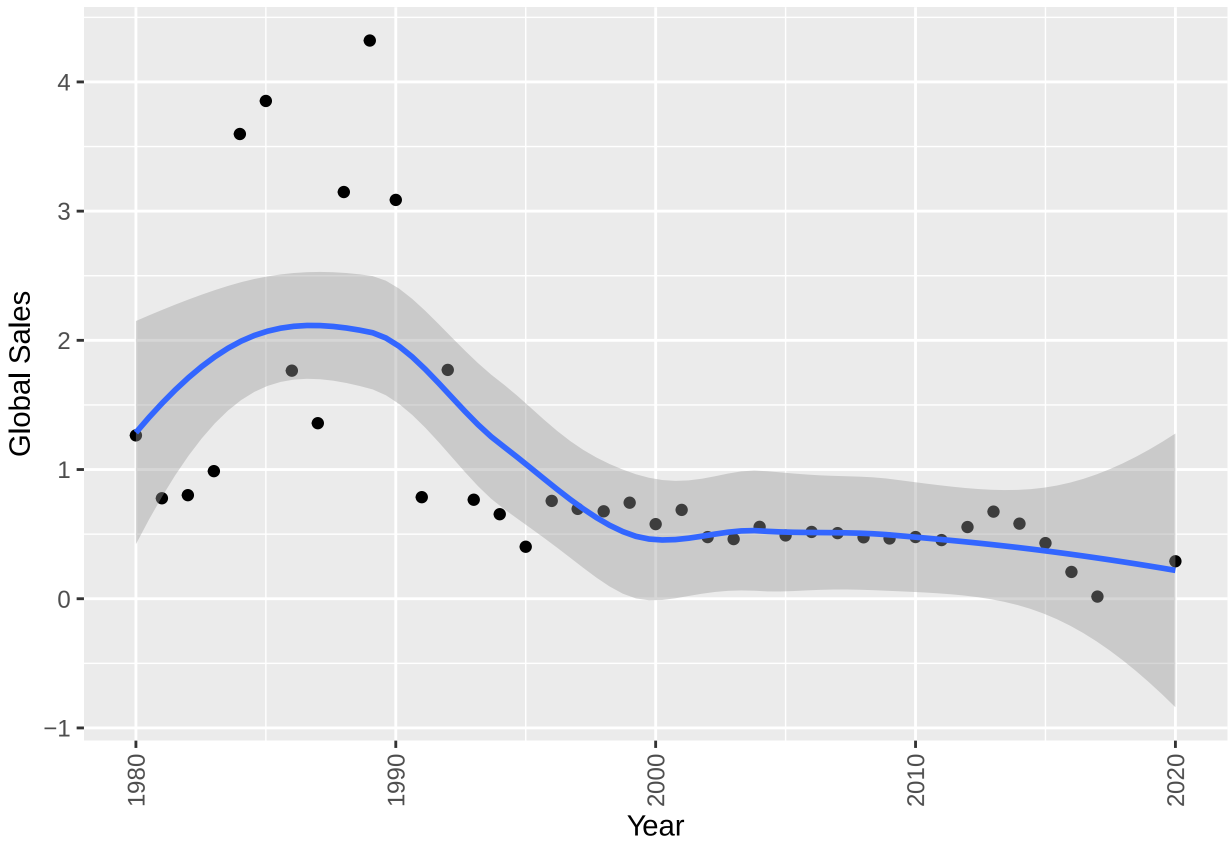
Okay, that's a lot of data. There's spikes during some years, but it's difficult to tell what's happening. Let's visualize the sales better by taking the average global sales of each year!

```
# find the average of the 6th column ~ Global Sales across the years
avg_data <- aggregate(tidy_data[6], list(tidy_data$Year), mean)
# renaming from Group.1 to Year appropriately
names(avg_data)[1] <- "Year"
head(avg_data)
```

```
##   Year Global_Sales
## 1 1980    1.2644444
## 2 1981    0.7776087
## 3 1982    0.8016667
## 4 1983    0.9876471
## 5 1984    3.5971429
## 6 1985    3.8528571
```

With the data above, we can easily visualize the trend of global sales throughout the year. Let's plot this new data frame and see what happens!

```
# mutating the year to the appropriate numeric type
avg_data <- avg_data %>%
  mutate(Year = as.numeric(as.character(Year)))

avg_plot <- avg_data %>%
  ggplot(mapping = aes(x = (Year), y = Global_Sales)) +
  geom_point () +
  geom_smooth(method='loess', formula=y~x)
avg_plot +
  xlab("Year") + ylab("Global Sales") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5))
```



This is much better to look at! We can see a huge spike in video games sales between the years of 1981-1992. The years prior to that spike remains relatively "stable", hovering around the .5 mark in global sales. Interestingly, the lowest average video game global sales happend in 2017.

Global Sales vs. Different Genres

We now want to examine the global sales in terms of different genres. With this data we can answer certain questions, like which genres produce the most revenue, or which genres are the most popular over time.

The first thing we must do is sum the global sales grouped by the genre. This will allow us to summarize which genres people have been more popular over time

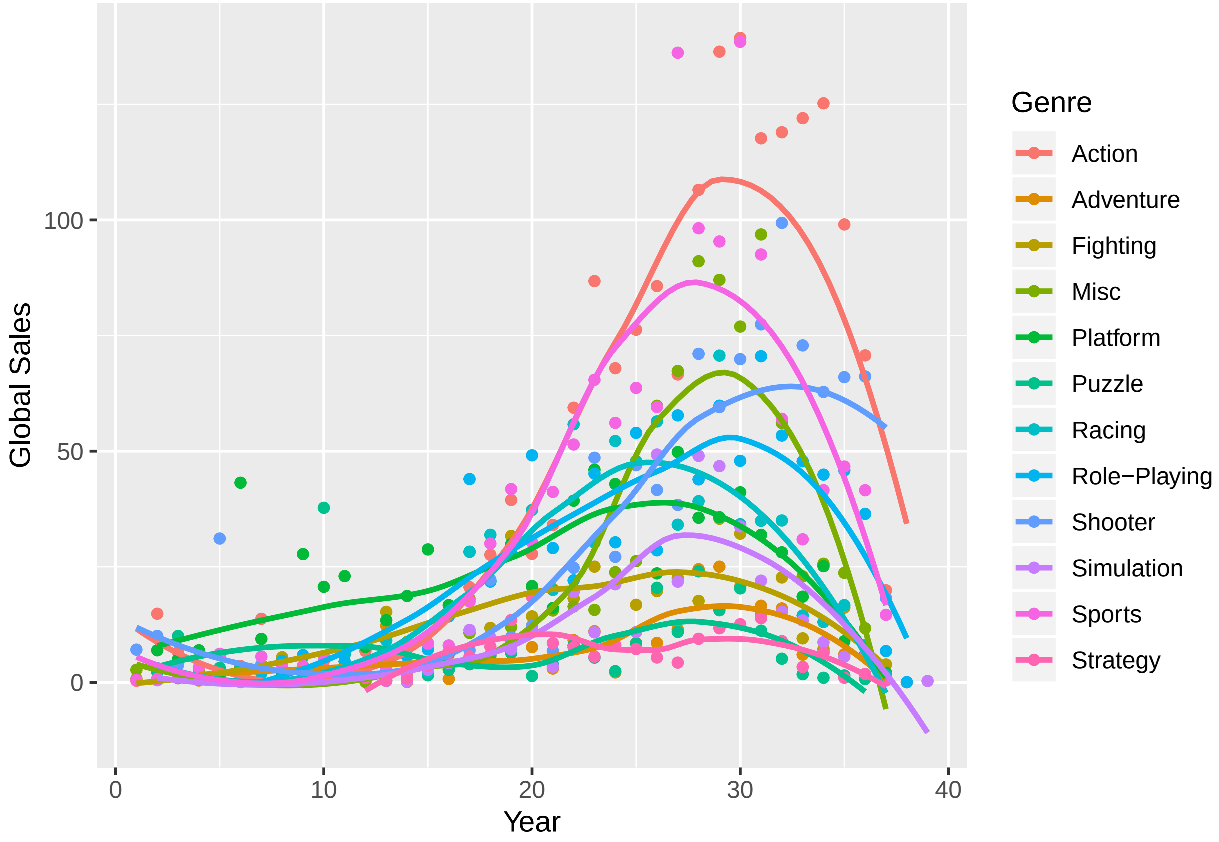
```
sales_by_genre <- aggregate(Global_Sales~Genre, tidy_data, sum)
head(sales_by_genre)
```

```
##       Genre Global_Sales
## 1    Action      1722.84
## 2 Adventure       234.59
## 3  Fighting       444.05
## 4      Misc       789.87
## 5  Platform       829.13
## 6    Puzzle       242.21
```

So this is interesting. It seems action-games are the most popular genres, followed by sports-games and then shooters. But this just summarizes the data taken as a whole, something more insightful might be to visualizethe popularity of each genre over time. To do this we're going to

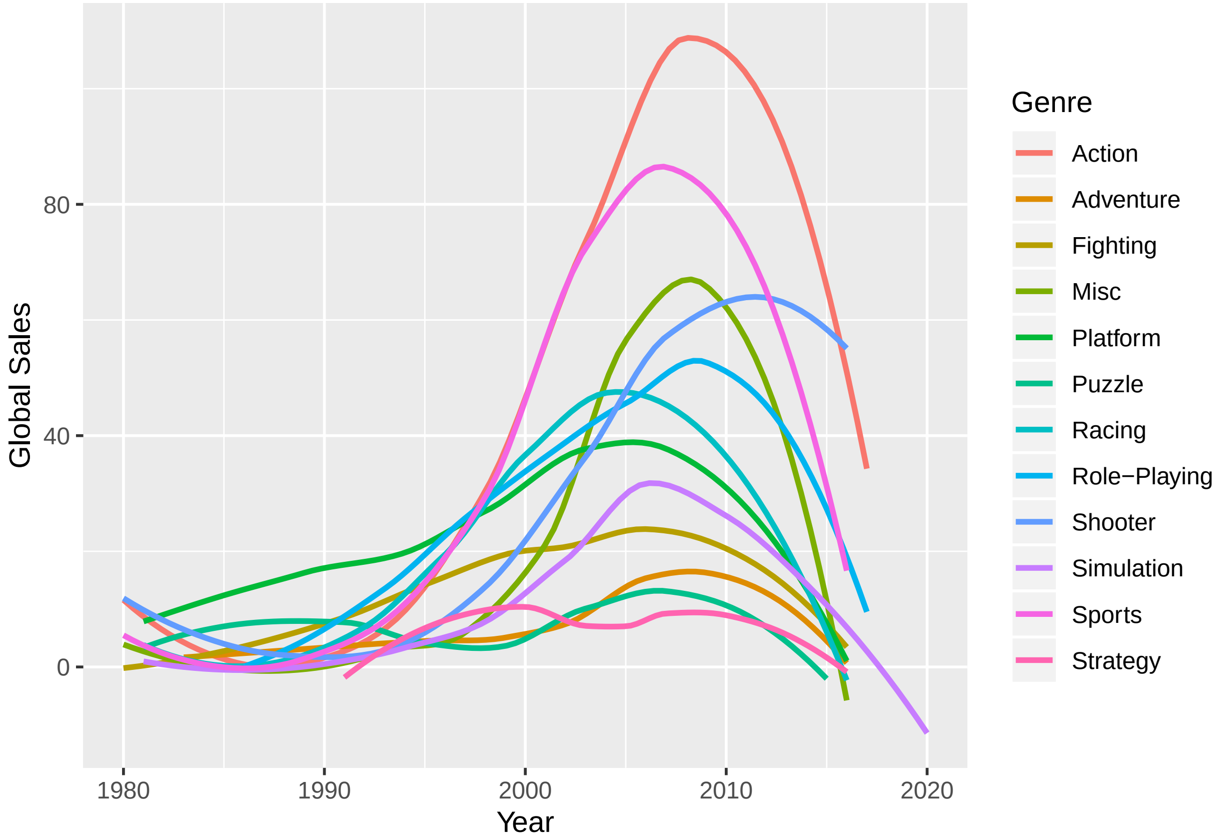
1. summarize the data's global sales in terms of Genre and Year
2. graph the data grouped by it's genre

```
sales_by_genre_and_time <- aggregate(Global_Sales~Genre+Year,
                                     tidy_data, sum) %>%
# must mutate the data into non-factors to be graphed
mutate(Year = as.numeric(Year)) %>%
mutate(Genre = as.character(Genre))

genre_time_plot <- sales_by_genre_and_time %>%
  ggplot(mapping = aes(x=Year, y=Global_Sales, color=Genre)) +
  geom_point() +
  geom_smooth(method='loess', formula=y~x, se=F)
genre_time_plot +
  xlab("Year") + ylab("Global Sales")
```



This plot is interesting, but's an absolute mess. Furthermore, our years are off now. Why is that? Well when you cast a factor like the Year to a number, it converts it into the magnitude of the factor instead of behaving as we would expect. To solve this, we will cast our Year variable to a character first, then cast that to an integer. To make our plot more readable, we will remove all of the data points and just make this a line graph.

```
sales_by_genre_and_time <- aggregate(Global_Sales~Genre+Year,
                                     tidy_data, sum) %>%
# convert the Years into characters, then to numbers
mutate(Year = as.numeric(as.character(Year))) %>%
mutate(Genre=as.character(Genre))

genre_time_plot <- sales_by_genre_and_time %>%
  ggplot(mapping = aes(x=Year, y=Global_Sales, color=Genre)) +
  geom_smooth(method='loess', formula=y~x, se=F)

genre_time_plot +
  xlab("Year") + ylab("Global Sales")
```



This is a much cleaner graph than the previous one, and there's much less extraneous data. As can be seen, the action genre is the most popular from a holistic view, and it's popularity has consistently grown over time, followed closely by the sports genre. However, in the late 2010's, shooter games rose to the top in grossing!

Global Sales vs. Different Publishers

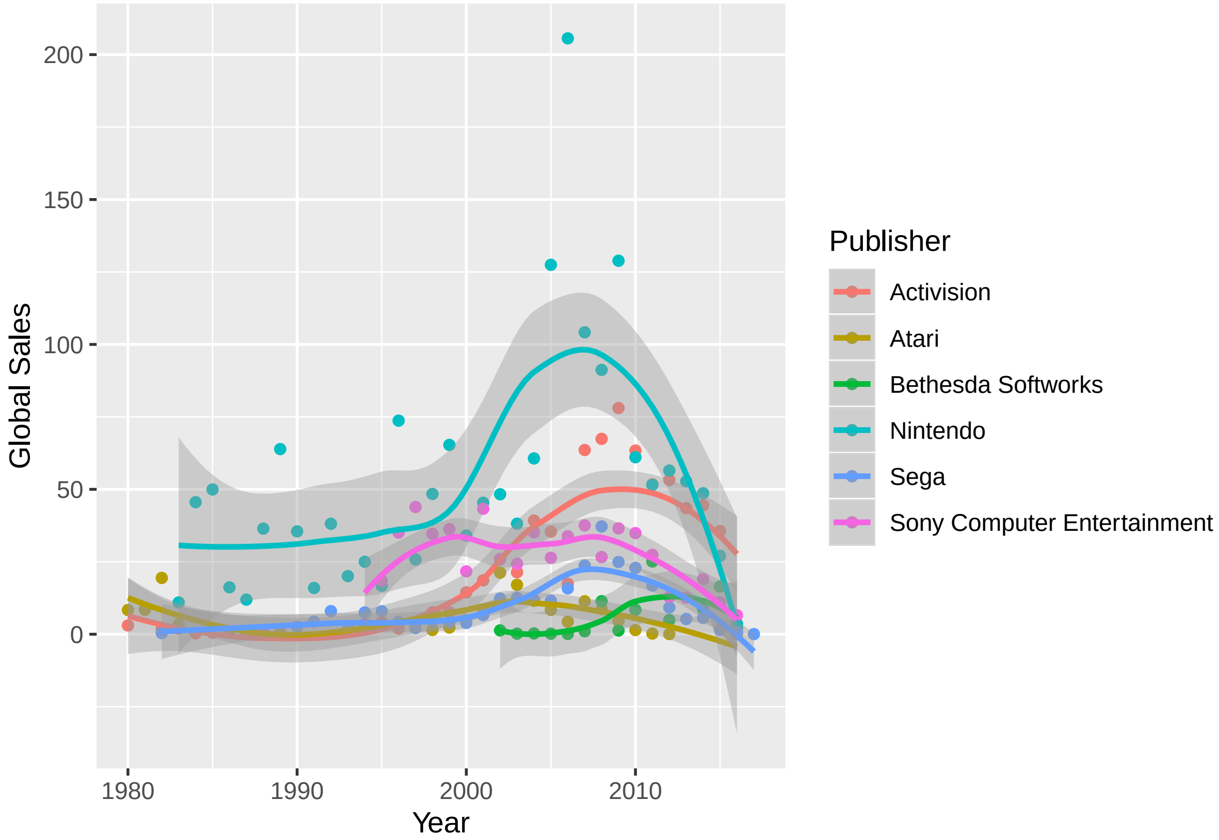
Now we wish to analyize sales by publisher. Let's begin by analyzing sales over time for a select few publishers.

```
# filtering our desired data
targets = c("Bethesda Softworks", "Atari", "Activision",
            "Nintendo", "Sony Computer Entertainment", "Sega")

sales_pub <- aggregate(Global_Sales~Year+Publisher, tidy_data, sum) %>%

# convert the Years into characters, then to numbers
mutate(Year = as.numeric(as.character(Year))) %>%
mutate(Publisher=as.character(Publisher))

sales_pub <- sales_pub %>%
filter(Publisher %in% targets)
head(sales_pub)
```

```
##   Year  Publisher Global_Sales
## 1 1980 Activision         3.02
## 2 1981 Activision         8.50
## 3 1982 Activision         1.86
## 4 1983 Activision         1.94
## 5 1984 Activision         0.27
## 6 1985 Activision         0.48
```

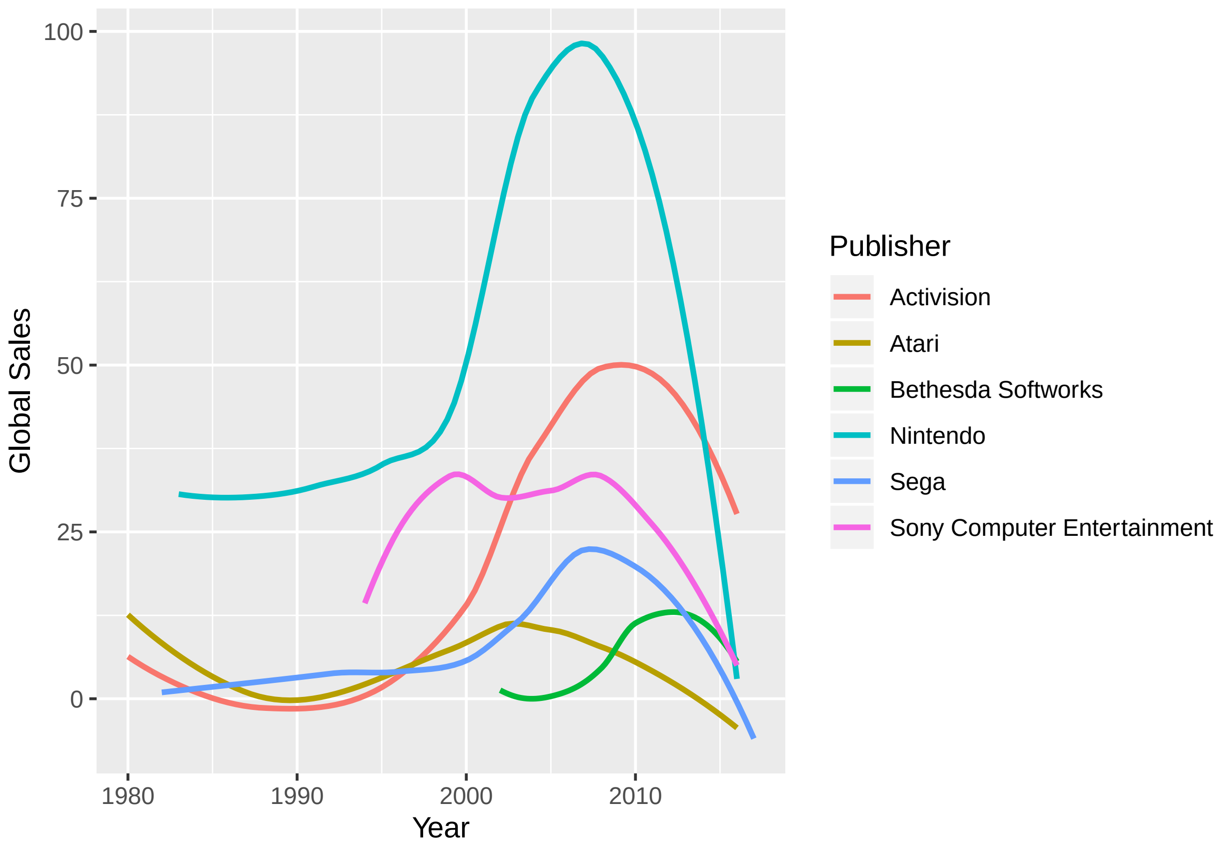
Now that we have our desired filtered data, let's visualize this data!

```
pub_time_plot <- sales_pub %>%
  ggplot(mapping = aes(x=Year, y=Global_Sales, color=Publisher)) +
  geom_point() +
  geom_smooth(method ='loess')
pub_time_plot +
  xlab("Year") + ylab("Global Sales")
```



That already looks very good. However, let's remove these dots and solely focus on the trends of these publisher!

```
pub_time_plot <- sales_pub %>%
  ggplot(mapping = aes(x=Year, y=Global_Sales, color=Publisher)) +
  geom_smooth(method='loess', formula=y~x, se=F)
pub_time_plot +
  xlab("Year") + ylab("Global Sales")
```



From this graph, Nintendo holistically has been at the top of video game publisher when it comes to global sales. However, Activision rose to the top as of recent times (which corresponds to the rise of shooting games since Activision is known for First-Person-Shooter games).

Step 4: Analysis, Hypothesis Testing and Machine Learning

Hypothesis Testing

Hypothesis testing is an essential procedure in statistics. A hypothesis test evaluates two mutually exclusive statements about a population to determine which statement is best supported by the sample data. **Hypothesis Testing** uses the important mathematical concepts like the *Law of Large Numbers* and *Central Limit Theorem* to decipher whether your hypothesis is the case or not.

Let's think about what to hypothesize within our data! Looking at the history of video games and the major events within its short 40 years time line, the 2 major events for console gaming was: the introduction of game cartridge vs. CD's. The introduction of game cartridge was popularized via Nintendo Entertainment System, or NES. The NES was released in 1985. On the other hand, CD's was popularized via the Playstation, or PS1. The PS1 was released in 1995. Both are groundbreaking events in the world of videogames! Which was more popular (measured via global sales)? With that in mind, that will be our hypotheses.

Null Hypothesis: The release of PS1 (or the introduction of gaming CDs) received more global sales
Alternative Hypothesis: The release of the NSE (or the introduction of gaming cartridges) received more global sales

We will use a *paired t-test* to test our hypothesis. The paired samples t-test is used to compare the means between two related groups of samples. As regards to our rejection value, we will reject our null hypothesis if our calculated p-value is less than 0.05

```r
# filtering our all of the PS and NES games
ps_vec <- filter(tidy_data, Platform == "PS")$Global_Sales
nes_vec <- filter(tidy_data, Platform == "NES")$Global_Sales

ps_vec <- sort(ps_vec, decreasing = TRUE)
# since NES only has 98 entries, we filtered out the top 98 grossed games of PS
ps_vec <- ps_vec[1:98]
nes_vec <- sort(nes_vec, decreasing = TRUE)

# performating the t-test
t_test <- t.test(ps_vec, nes_vec, paired = TRUE, alternative = "greater")
# divide by 2 because one-tailed t-test
t_test$p.value/2
```

```
## [1] 0.01013816
```

Since we are performing a *one-tailed t-test*, we would need to divide the p value in half. A one-tailed test allots all of your alpha to testing the statistical significance in the one direction of interest. In this case, we are only testing whether or not the release of PS1 gathered more income than the release of NES, or vise versa; but not **both**.

Based on our hypothesis testing, our *p* value is 0.01013816, which is less than 0.05. Therefore, we reject the null hypothesis. Our data suggests that the amount of global sales gathered from the introduction of the PS1 is *greater* than the amount with NES.

Machine Learning

Analysis and Machine Learning is a very important topic within data science. A precise prediction of said values can be of huge importance in many different aspects. In our case, let's say we want to create a program to estimate the global sales of your game. One can achieve this through the power of Machine Learning! You can train a model to look at the attributes of the game and determine it's probable sales.

The first thing we need to do is to define our data in terms of one of two values. What I'm going to do is standardize the data and create a new variable that is "Above Average" or "Below Average", dependent on whether the global sales were at least average for their year. We'll only take a sample of the data, about 5% of it. This is a large enough sample that the accuracy of the model will be guaranteed by the *Central Limit Theorem.*

To standardize the data, we create a Z_Sales variable that equals the Global_Sales minus the mean sales for that year, divided by the standard deviation. This will display our data in terms of distance from the mean. So a Z_Sales value 0 means that the data is precisely average for that year. A Z_Sales value of 1 means the sales for that game is 1 standard deviation above the average for that year.

```r
standard_df <- tidy_data %>%
  mutate(Z_Sales = NA)

for(y in 1980:2020) {
  stdev <- sd(filter(standard_df, Year==y)$Global_Sales)
  avg <- mean(filter(standard_df, Year == y)$Global_Sales)
  standard_df <- standard_df %>%
    mutate(Z_Sales = ifelse(Year == y, ((Global_Sales - avg)/stdev), Z_Sales))
}
# a sample size of 815 data points from the 16,000 total points
standard_df <- standard_df[sample(nrow(standard_df), 815),]
```

Next we want to categorize our data. We will denote the Success_Level as “Above Average” or “Below Average” in terms of Z_Sales. So if a game sold above or equal to the global average for that year, which is signified by it’s Z_Sales being greater than or equal to zero, then we will mark it as above average. If a game sold below average, which means it’s Z_Sales variable is below zero, then we will denote as below average appropriately.

We will then remove the Global_Sales, Z_Sales, and Name attributes from our data, because they are no longer relevant to our analysis. Although a game’s name might affect it’s sales, R has no way of knowing how (How do you tell a program that a “cool-sounding” game will sell better? What does a program think a “cool-sounding” name is?). We remove the Z_Sales and the Global_Sales because these are the things we are predicting. Leaving them in is like giving the model the answer.

```r
standard_df <- standard_df %>%
  mutate(Success_Level = ifelse(Z_Sales >= 0, 'Above Average', 'Below Average')) %>%
  select(-Global_Sales, -Name, -Z_Sales)
head(standard_df)
```

```
##   Platform Year        Genre              Publisher Success_Level
## 1      3DS 2012         Misc     Avanquest Software Below Average
## 2       DS 2008 Role-Playing               Nintendo Above Average
## 3     SNES 1990       Action                 Capcom Below Average
## 4      PSP 2010         Misc         Ubisoft Annecy Below Average
## 5      PS2 2002       Sports Zoo Digital Publishing Below Average
## 6     X360 2012      Shooter                Ubisoft Above Average
```

The next thing we need to do is divide our data into a training set and a testing set. We will use the training set to train our model, then test our model on the testing set to determine it’s accuracy. We will use the k-fold cross-validation algorithm to train our data. That means we will divide our data into 10 sets, or 10 “folds”. We will use 9 of those folds as the training data, and we will use the final fold for validation, our testing set. That sounds complicated, but the details are abstracted away and wrapped in the createFolds function.

```r
set.seed(1234)

partitionRule <- createFolds(standard_df$Success_Level, k=10, list=F)
trainingSet <- standard_df[partitionRule,]
testingSet <- standard_df[-partitionRule,]

# we have to rename the columns after splitting them up
names(trainingSet) <- make.names(colnames(trainingSet))
names(testingSet) <- make.names(colnames(testingSet))
```

Now we build the model. We will be using a K-Nearest-Neighbor model for our predictions because it’s a very simple model to work with because it’s not dependent on the distribution of the original data. It basically just graphs the Success_Level in terms of everything else and compares how close these points are on the graph. This is what the “as.factor(Success_Level)~.” formula tells the train function to do, where the as.factor(Success_Level) essentially makes the Success_Level the dependent variable, and the . means to graph this variable in terms of every other variable.

As a side note, if we used different model, we would have to pass different arguments to the train and predict functions that are relevant to those functions specifically. Since we’re just using the KNN model, we don’t require any extra parameters, but it should be noted that the KNN model does use a lot of extra memory space.

```
knn_fit <- train(as.factor(Success_Level)~., data=trainingSet, method='knn')
knn_predict <- predict(knn_fit, newdata=testingSet)
confusionMatrix(knn_predict, as.factor(testingSet$Success_Level))
```

```
## Confusion Matrix and Statistics
##
##                Reference
## Prediction      Above Average Below Average
##   Above Average            75           167
##   Below Average           111           452
##
##                Accuracy : 0.6547
##                  95% CI : (0.6207, 0.6875)
##     No Information Rate : 0.7689
##     P-Value [Acc > NIR] : 1.0000000
##
##                   Kappa : 0.1207
##
##  Mcnemar's Test P-Value : 0.0009714
##
##             Sensitivity : 0.40323
##             Specificity : 0.73021
##          Pos Pred Value : 0.30992
##          Neg Pred Value : 0.80284
##              Prevalence : 0.23106
##          Detection Rate : 0.09317
##    Detection Prevalence : 0.30062
##       Balanced Accuracy : 0.56672
##
##        'Positive' Class : Above Average
##
```

Step 5: Analysis

Here are the information gathered from the steps above:

- Huge spike of video global sales in the period of 1981-1992. The average grossing peaked at the late 90's. The years after that initial spike remained relative static.
- From a holistic view, action games are undoubtedly the most popular amongst the variety of other genres. However, in the late 2010's, shooter game rose to the top in terms of grossing.
- From a holistic view, Nintendo has been at the top of video game publisher when it comes to global sales. However, Activision rose to the top as of recent times (which corresponds to the rise of shooting games since Activision is known for First-Person-Shooter games).
- The first release of gaming CDs via PS1 gathered more interest/popularity (measured by global sales) than the first release of gaming cartridges via NES.
- A ML model can be constructed that predicts the grossing of a game based on it's publisher, the year of it's released, it's genre, and it's platform.

From those information gathered, one can synthesize that: the global sales peaked in the late 90s, and thus the introduction of CD-ROM was also in the late '90s (popularized by PS1 in 1995). Therefore, the introduction of CDs possibly aided the peaked grossing in the late '90s. This would make sense due to the increase in accessibility for the customer. Furthermore, gaming CD's are still in use in today's video gaming world, which demonstrates the long lasting effect of this invention. Therefore, one could say that another revolutionary change in the way we play games could spark another spike in video game grossing (something to the extent of CD-ROM). Recently, cloud gaming and digital distribution unsurpringly seems to be gaining popularity due to convenience purporses. Will it cause another spike in video game grossing? We shall see :)

Moreover, FPS or shooter games seems to be the highest grossing genre as of recent time. With its high demands, aspiring game makers could potentially exploit more on this genre (with other factors demonstrated from the ML model), and thus earn a lot of revenue!

That concludes our tutorial. Thank you so much for making it all the way through this tutorial! We hope that you learned something new from this, and start your own data adventure sometime soon :)