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("") in your RStudio's console. Replace the 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

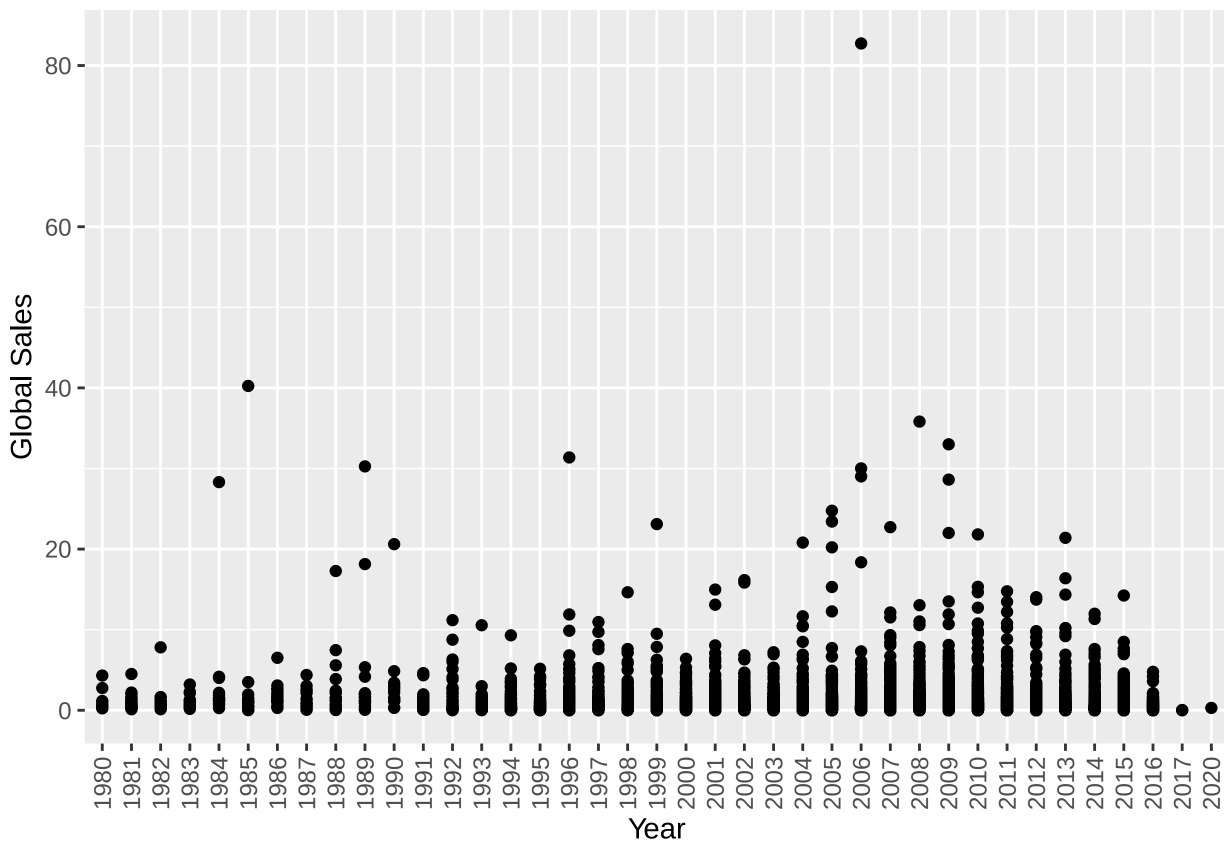
Exploratory Data Analysis is the last step before statistical analysis and machine learning. In this steps, we will cover the basics of visualizing datas through different graphing techniques and applications of linear regressions. Furthermore, yt is important to spot nuances like skew, the distribution of data, any sort of trends, how pairs of variables interact, and any problems with the data. The 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!

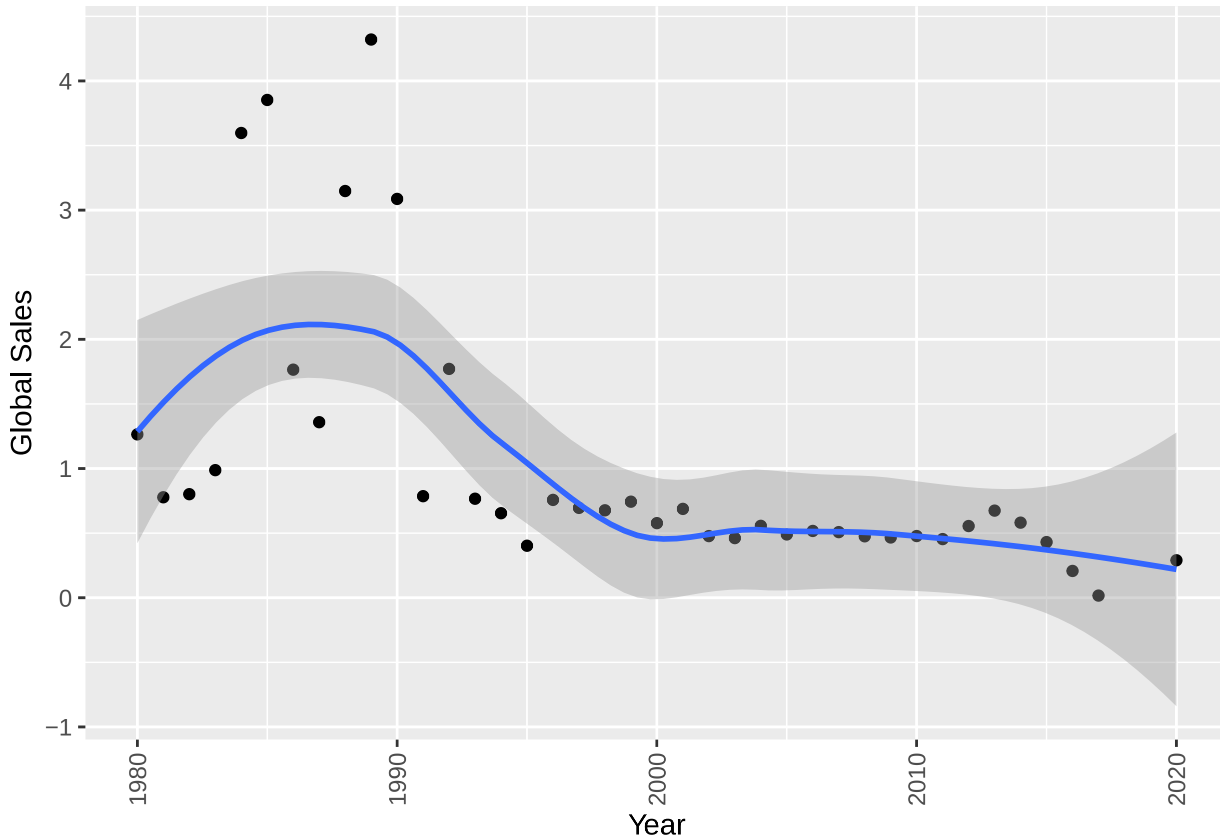
```
# find the average of the 6th column ~ Global Sales across the years
avg_data <- aggregate(tidy_data[6], list(tidy_data$Year), mean)
# renaming from Group.1 to Year appropriately
names(avg_data)[1] <- "Year"
head(avg_data)
```

```
##   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)
```

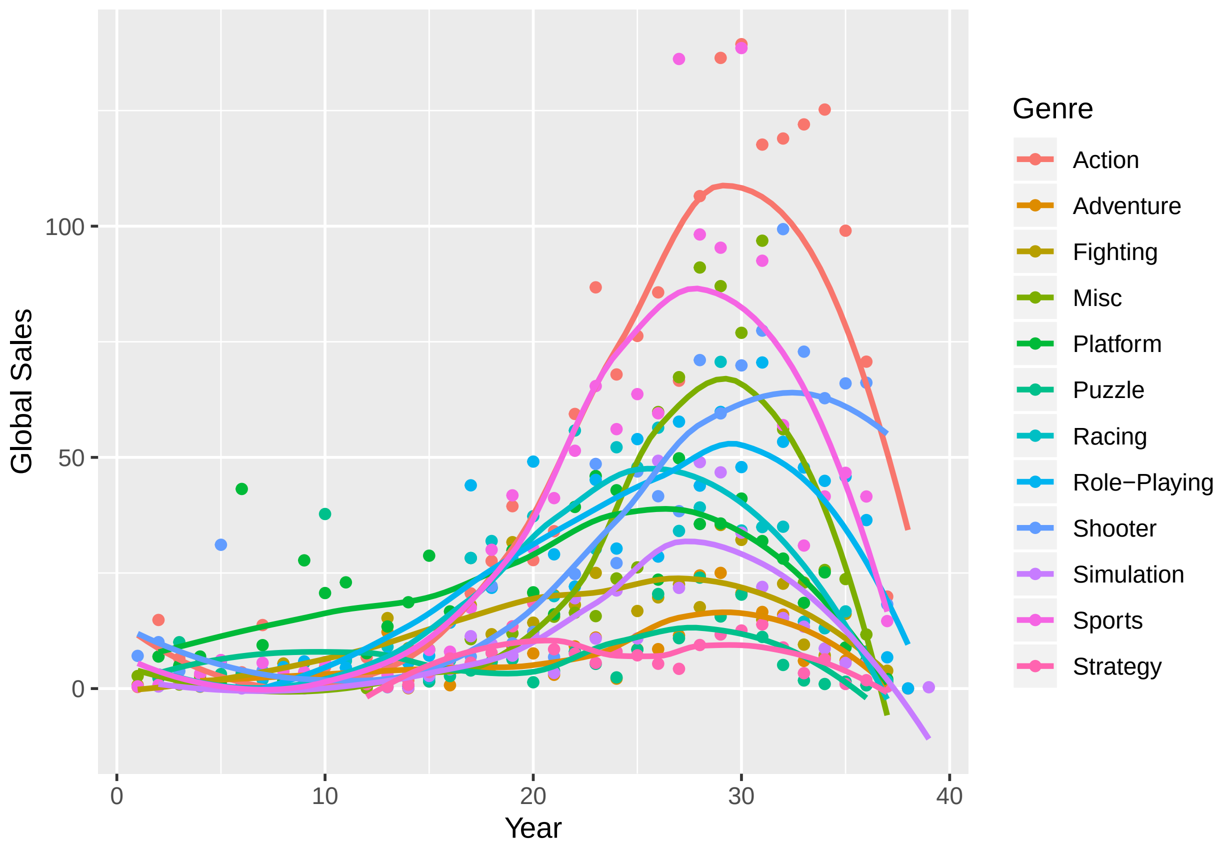
```
##       Genre Global_Sales
## 1    Action      1722.84
## 2 Adventure       234.59
## 3  Fighting       444.05
```

```
## 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

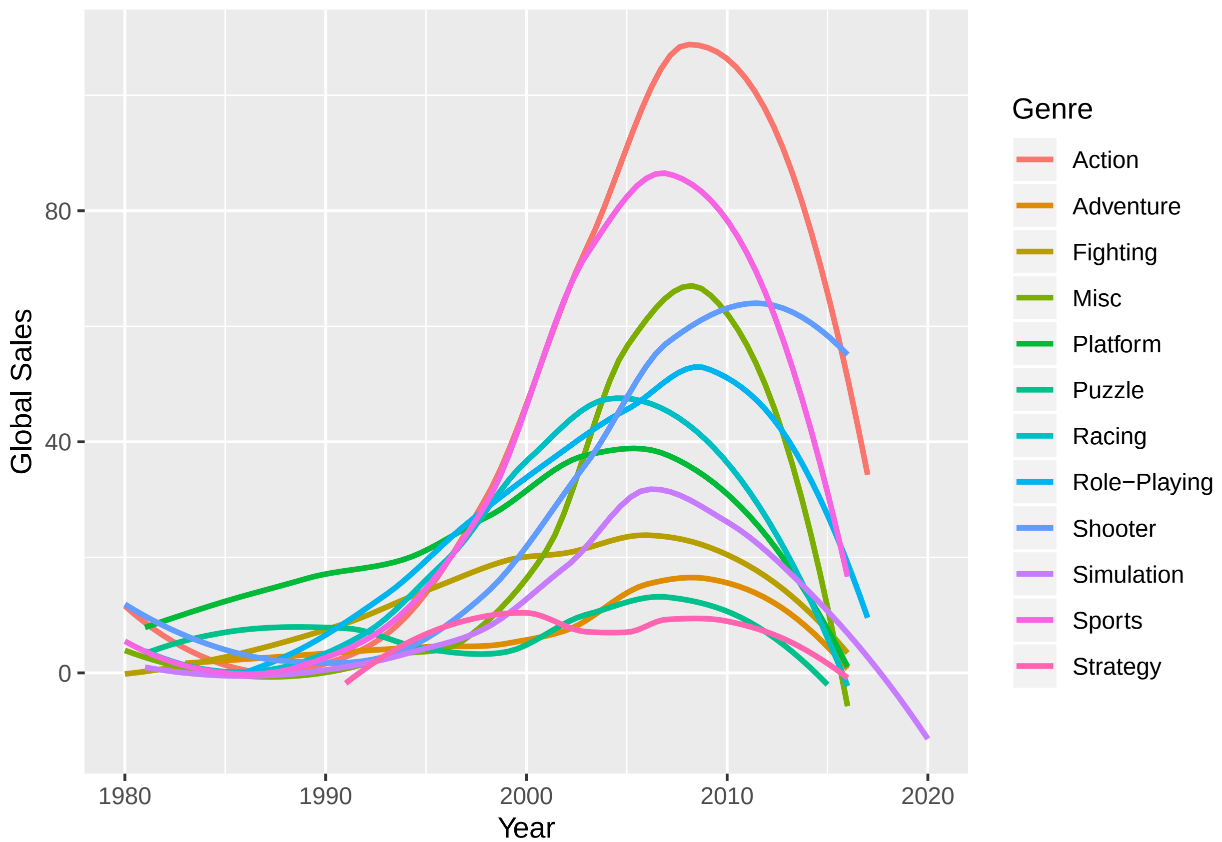
1. summarize the data's global sales in terms of Genre and Year
2. graph the data grouped by it's genre

```
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")
```



As can be seen, the action genre is the most popular, and it's popularityhas consistently grown over time, followed closely by the sports genre.

Global Sales vs. Different Publishers

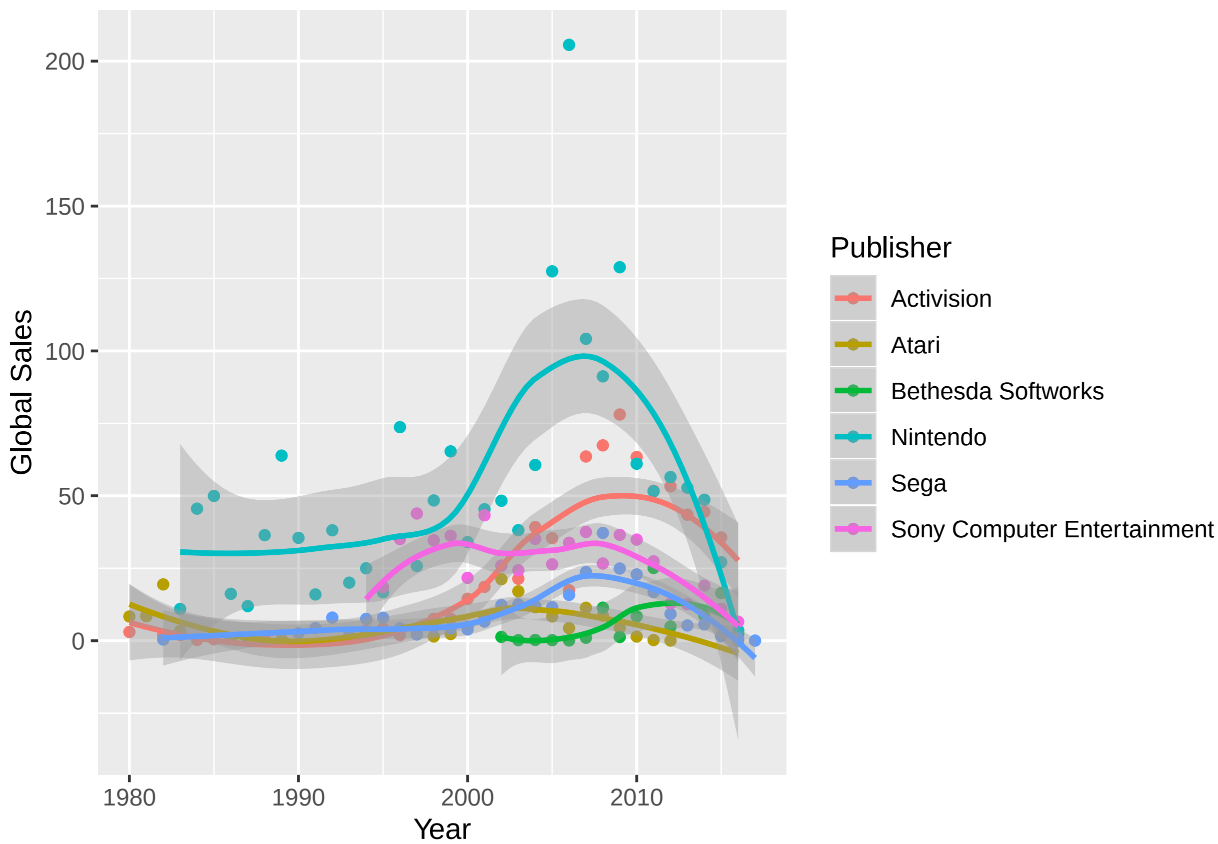
Now we wish to analyize sales by publisher. Let's begin by analyzing sales over time for a select few publishers.

```
# 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
```

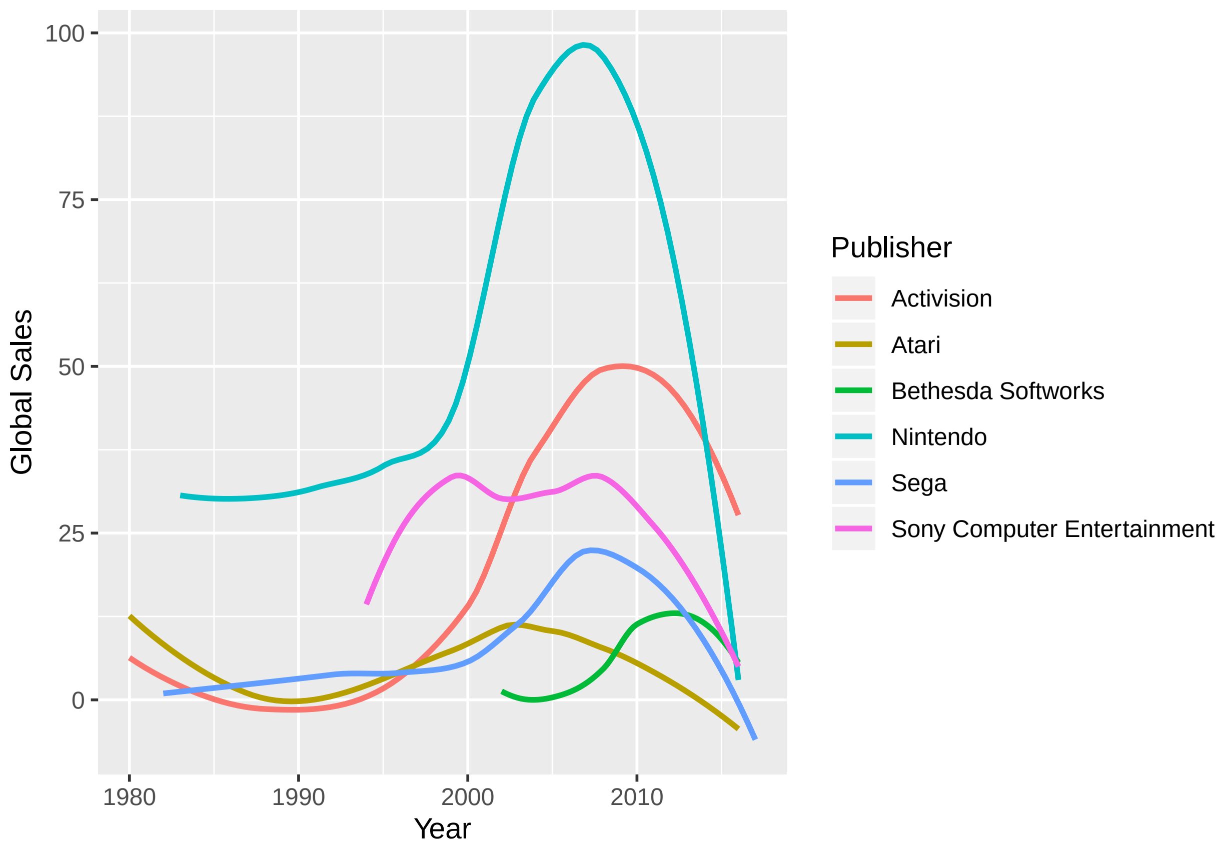
Now that we have our desired filtered data, let's visualize this data!

```
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 delete 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, we can see that 4 out of the 6 publishers all had a similar trend in regards to global sales over the years. Activision, Sega, Nintendo, and Bethesda Softworks all had a huge peaked then plumitted in sales. Atari was the sole publisher that remained a relatively constant trend. Sony had an increased in sale and remained in the late 90's and remained relatively constant until the 2010 ish and plumitted back down.

Step 4: Analysis, Hypothesis Testing and Machine Learning

So now let's say we want to create a program to estimate the global sales of a game. The way to do this is to train a model such that it can 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 True or False 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*.

```
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),]
head(standard_df)
```

```
##                                          Name Platform Year        Genre
## 15072 Casper's Scare School: Spooky Sports Day      Wii 2009       Sports
## 11871    Dynasty Warriors DS: Fighter's Battle       DS 2007       Action
## 9305              Shrek Smash n' Crash Racing       GC 2006       Racing
## 14750                        Fishdom 2 Deluxe       PC 2010       Puzzle
## 3902                  Monster Hunter Freedom 2      PS3 2011 Role-Playing
## 14631                             R-Type Delta       PS 1998      Shooter
##                       Publisher Global_Sales     Z_Sales
## 15072  Blast! Entertainment Ltd         0.02 -0.28522588
## 11871                Tecmo Koei         0.07 -0.35862103
## 9305                 Activision         0.13 -0.12714766
## 14750                Rondomedia         0.02 -0.35300628
## 3902                     Capcom         0.50  0.04016399
## 14631 Irem Software Engineering         0.03 -0.51723758
```

Next we want to categorize our data. We will denote the Success_Level as above average or below average in terms of global 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

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

```
##   Platform Year        Genre                 Publisher     Z_Sales
## 1      Wii 2009       Sports  Blast! Entertainment Ltd -0.28522588
## 2       DS 2007       Action                Tecmo Koei -0.35862103
## 3       GC 2006       Racing                Activision -0.12714766
## 4       PC 2010       Puzzle                Rondomedia -0.35300628
## 5      PS3 2011 Role-Playing                    Capcom  0.04016399
## 6       PS 1998      Shooter Irem Software Engineering -0.51723758
##   Success_Level
## 1 Below Average
## 2 Below Average
## 3 Below Average
## 4 Below Average
## 5 Above Average
## 6 Below 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.

```
set.seed(1234)

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

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.

```
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            54            43
##   Below Average           116           592
##
##                Accuracy : 0.8025
##                  95% CI : (0.7733, 0.8295)
##     No Information Rate : 0.7888
##     P-Value [Acc > NIR] : 0.1827
##
##                   Kappa : 0.2966
##
##  Mcnemar's Test P-Value : 1.13e-08
##
##             Sensitivity : 0.31765
##             Specificity : 0.93228
##          Pos Pred Value : 0.55670
##          Neg Pred Value : 0.83616
##              Prevalence : 0.21118
##          Detection Rate : 0.06708
##    Detection Prevalence : 0.12050
##       Balanced Accuracy : 0.62497
##
##        'Positive' Class : Above Average
##
```

b