Final\_Project\_Q3.1\_R\_Script.R

2025-05-03

***## RESEARCH QUESTION 3.1 R-SCRIPT***  
**library**(tidyverse)

## Warning: package 'tidyverse' was built under R version 4.3.3

## Warning: package 'tidyr' was built under R version 4.3.3

## Warning: package 'readr' was built under R version 4.3.3

## Warning: package 'purrr' was built under R version 4.3.3

## Warning: package 'dplyr' was built under R version 4.3.3

## Warning: package 'stringr' was built under R version 4.3.3

## Warning: package 'forcats' was built under R version 4.3.3

## Warning: package 'lubridate' was built under R version 4.3.3

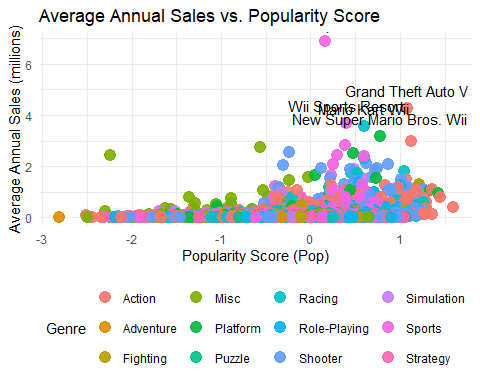
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──  
## ✔ dplyr 1.1.4 ✔ readr 2.1.5  
## ✔ forcats 1.0.0 ✔ stringr 1.5.1  
## ✔ ggplot2 3.5.2 ✔ tibble 3.2.1  
## ✔ lubridate 1.9.3 ✔ tidyr 1.3.1  
## ✔ purrr 1.0.2   
## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ dplyr::lag() masks stats::lag()  
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors

*# Read in the data*  
data <- **read.csv**("C:/Users/slhor/Downloads/vgsales\_cleanfullmodel.csv")  
data <- data **%>%**  
 **filter**(**!is.na**(Year))  
data <- data **%>%**  
 **mutate**(Year = **as.numeric**(Year)) **%>%**  
 **mutate**(avg\_annual\_sales = Global\_Sales **/** (2018 **-** Year))

## Warning: There was 1 warning in `mutate()`.  
## ℹ In argument: `Year = as.numeric(Year)`.  
## Caused by warning:  
## ! NAs introduced by coercion

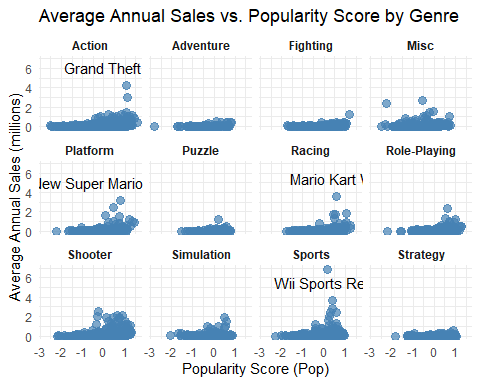
***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
*# Scatter plot for avg annual sales and Pop Score*  
**library**(ggplot2)  
**library**(dplyr)  
  
*# Find the 5 games with the highest avg\_annual\_sales*  
top\_games <- data **%>%**  
 **arrange**(**desc**(avg\_annual\_sales)) **%>%**  
 **head**(5)  
  
*# Create the plot with points colored by Genre*  
**ggplot**(data, **aes**(x = Pop, y = avg\_annual\_sales, color = Genre)) **+**  
 **geom\_point**(alpha = 0.9, size = 4) **+** *# Increased size of the points*  
   
 *# Add labels for the top 5 games with the highest avg annual sales*  
 **geom\_text**(data = top\_games, **aes**(label = Name), vjust = **-**1, hjust = 0.5, color = "black") **+**  
   
 **labs**(  
 title = "Average Annual Sales vs. Popularity Score",  
 x = "Popularity Score (Pop)",  
 y = "Average Annual Sales (millions)"  
 ) **+**  
 **theme\_minimal**() **+**  
 **theme**(legend.position = "bottom") *# To adjust the position of the legend*

## Warning: Removed 75 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Separate Scatter plots for Pop and Sales over genre***  
**library**(ggplot2)  
**library**(dplyr)  
  
*# Find the 5 games with the highest avg\_annual\_sales*  
top\_games <- data **%>%**  
 **arrange**(**desc**(avg\_annual\_sales)) **%>%**  
 **head**(5)  
  
*# Create faceted scatter plots by Genre*  
**ggplot**(data, **aes**(x = Pop, y = avg\_annual\_sales)) **+**  
 **geom\_point**(color = "steelblue", alpha = 0.7, size = 3) **+** *# Single color for clarity*  
 **geom\_text**(data = top\_games, **aes**(label = Name), vjust = **-**1, hjust = 0.5, color = "black") **+**  
 **facet\_wrap**(**~** Genre) **+**  
 **labs**(  
 title = "Average Annual Sales vs. Popularity Score by Genre",  
 x = "Popularity Score (Pop)",  
 y = "Average Annual Sales (millions)"  
 ) **+**  
 **theme\_minimal**() **+**  
 **theme**(  
 strip.text = **element\_text**(face = "bold"),  
 legend.position = "none"  
 )

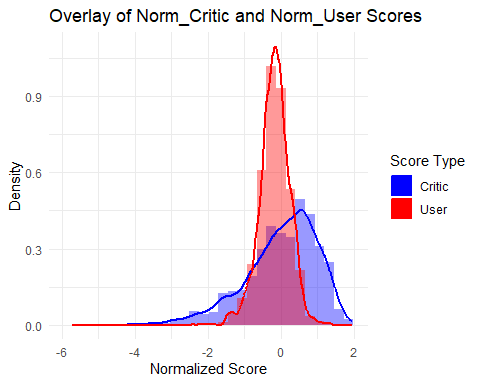
## Warning: Removed 75 rows containing missing values or values outside the scale range  
## (`geom\_point()`).



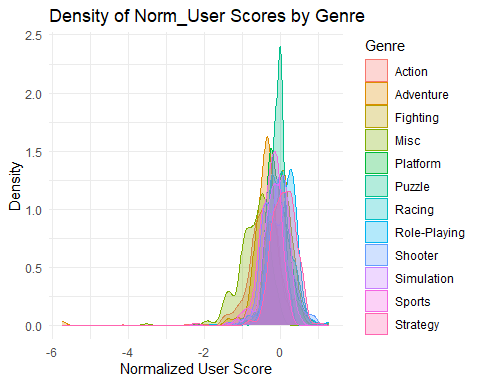
***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Histogram of Normalized User and Critic Scores***  
*# Load libraries*  
**library**(ggplot2)  
  
*# Create the plot with a legend*  
**ggplot**() **+**  
 **geom\_histogram**(data = data,   
 **aes**(x = Norm\_Critic, y = ..density.., fill = "Critic"),   
 alpha = 0.4, bins = 30) **+**  
 **geom\_histogram**(data = data,   
 **aes**(x = Norm\_User, y = ..density.., fill = "User"),   
 alpha = 0.4, bins = 30) **+**  
 **geom\_density**(data = data,   
 **aes**(x = Norm\_Critic, color = "Critic"), size = 1) **+**  
 **geom\_density**(data = data,   
 **aes**(x = Norm\_User, color = "User"), size = 1) **+**  
 **scale\_fill\_manual**(name = "Score Type", values = **c**("Critic" = "blue", "User" = "red")) **+**  
 **scale\_color\_manual**(name = "Score Type", values = **c**("Critic" = "blue", "User" = "red")) **+**  
 **labs**(title = "Overlay of Norm\_Critic and Norm\_User Scores",  
 x = "Normalized Score",  
 y = "Density") **+**  
 **theme\_minimal**()

## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
## ℹ Please use `linewidth` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.

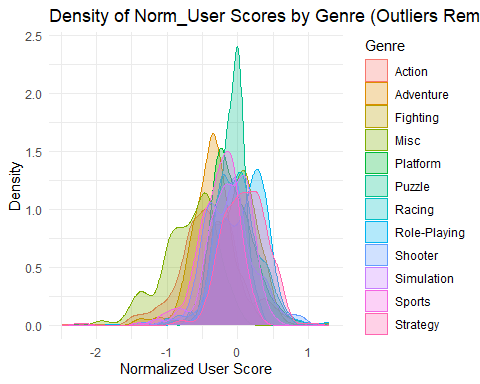
## Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.  
## ℹ Please use `after\_stat(density)` instead.  
## This warning is displayed once every 8 hours.  
## Call `lifecycle::last\_lifecycle\_warnings()` to see where this warning was  
## generated.



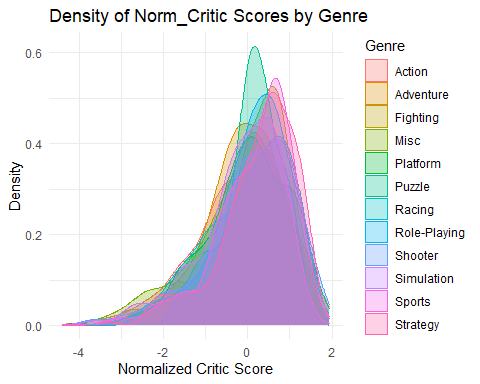
***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Histograms of Normalized User Scores (w/ and w/o Outliers)***  
**ggplot**(data, **aes**(x = Norm\_User, color = Genre, fill = Genre)) **+**  
 **geom\_density**(alpha = 0.3) **+**  
 **labs**(title = "Density of Norm\_User Scores by Genre",  
 x = "Normalized User Score",  
 y = "Density") **+**  
 **theme\_minimal**()



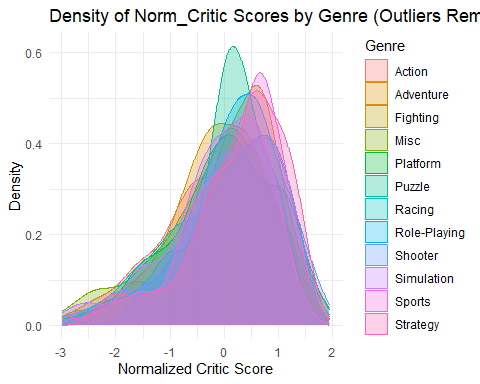
**library**(dplyr)  
**library**(ggplot2)  
  
*# Filter out extreme outliers for Norm\_User*  
filtered\_data\_user <- data **%>%**  
 **filter**(Norm\_User **>=** **-**3, Norm\_User **<=** 3)  
  
*# Create density plot without outliers*  
**ggplot**(filtered\_data\_user, **aes**(x = Norm\_User, color = Genre, fill = Genre)) **+**  
 **geom\_density**(alpha = 0.3) **+**  
 **labs**(title = "Density of Norm\_User Scores by Genre (Outliers Removed)",  
 x = "Normalized User Score",  
 y = "Density") **+**  
 **theme\_minimal**()



***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Histogram of Normalized Critic Scores (w/ and w/o Outliers)***  
**ggplot**(data, **aes**(x = Norm\_Critic, color = Genre, fill = Genre)) **+**  
 **geom\_density**(alpha = 0.3) **+**  
 **labs**(title = "Density of Norm\_Critic Scores by Genre",  
 x = "Normalized Critic Score",  
 y = "Density") **+**  
 **theme\_minimal**()



**library**(dplyr)  
**library**(ggplot2)  
  
*# Filter out extreme outliers for Norm\_Critic*  
filtered\_data\_critic <- data **%>%**  
 **filter**(Norm\_Critic **>=** **-**3, Norm\_Critic **<=** 3)  
  
*# Create density plot without outliers*  
**ggplot**(filtered\_data\_critic, **aes**(x = Norm\_Critic, color = Genre, fill = Genre)) **+**  
 **geom\_density**(alpha = 0.3) **+**  
 **labs**(title = "Density of Norm\_Critic Scores by Genre (Outliers Removed)",  
 x = "Normalized Critic Score",  
 y = "Density") **+**  
 **theme\_minimal**()



***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
*# One-way ANOVA for Critic Scores*  
anova\_critic <- **aov**(Norm\_Critic **~** Genre, data = data)  
**summary**(anova\_critic)

## Df Sum Sq Mean Sq F value Pr(>F)   
## Genre 11 90 8.143 8.313 1.3e-14 \*\*\*  
## Residuals 4157 4072 0.980   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

*# One-way ANOVA for user scores*  
anova\_user <- **aov**(Norm\_User **~** Genre, data = data)  
**summary**(anova\_user)

## Df Sum Sq Mean Sq F value Pr(>F)   
## Genre 11 200.7 18.243 129 <2e-16 \*\*\*  
## Residuals 4157 587.9 0.141   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Correlation Analysis***  
**library**(dplyr)  
  
*# Global correlation*  
cor\_global\_pop <- **cor**(data**$**Pop, data**$**Global\_Sales, use = "complete.obs", method = "pearson")  
**cat**("Global Correlation:**\n**")

## Global Correlation:

**cat**("Pop vs Global\_Sales:", cor\_global\_pop, "**\n\n**")

## Pop vs Global\_Sales: 0.2413045

*# Function to compute grouped correlations safely*  
compute\_group\_correlation <- **function**(df, group\_var, score\_var) {  
 df **%>%**  
 **group\_by**(.data[[group\_var]]) **%>%**  
 **summarise**(  
 n = **sum**(**!is.na**(.data[[score\_var]]) **&** **!is.na**(Global\_Sales)),  
 correlation = **if** (n **>=** 2) **cor**(.data[[score\_var]], Global\_Sales, use = "complete.obs", method = "pearson") **else** NA\_real\_  
 ) **%>%**  
 **filter**(**!is.na**(correlation)) **%>%**  
 **arrange**(**desc**(**abs**(correlation)))  
}  
  
*# Grouped correlations using "Pop"*  
cor\_genre\_pop <- **compute\_group\_correlation**(data, "Genre", "Pop")  
cor\_platform\_pop <- **compute\_group\_correlation**(data, "Platform", "Pop")  
cor\_publisher\_pop <- **compute\_group\_correlation**(data, "Publisher", "Pop")  
  
*# View top results*  
**cat**("Correlation by Genre:**\n**")

## Correlation by Genre:

**print**(**head**(cor\_genre\_pop))

## # A tibble: 6 × 3  
## Genre n correlation  
## <chr> <int> <dbl>  
## 1 Action 953 0.441  
## 2 Fighting 221 0.432  
## 3 Role-Playing 528 0.402  
## 4 Shooter 496 0.367  
## 5 Platform 326 0.349  
## 6 Racing 302 0.334

**cat**("**\n**Correlation by Platform:**\n**")

##   
## Correlation by Platform:

**print**(**head**(cor\_platform\_pop))

## # A tibble: 6 × 3  
## Platform n correlation  
## <chr> <int> <dbl>  
## 1 NES 3 0.691  
## 2 PSV 3 0.539  
## 3 N64 94 0.505  
## 4 GB 8 0.487  
## 5 XB 107 0.477  
## 6 PS4 38 0.467

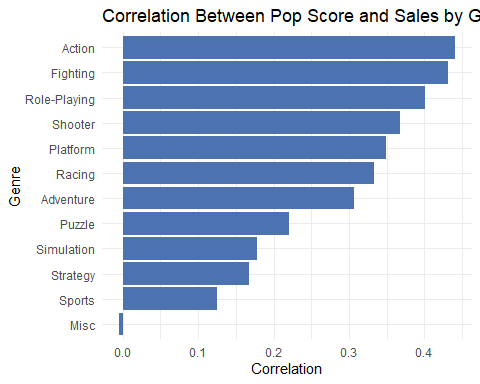
**cat**("**\n**Correlation by Publisher:**\n**")

##   
## Correlation by Publisher:

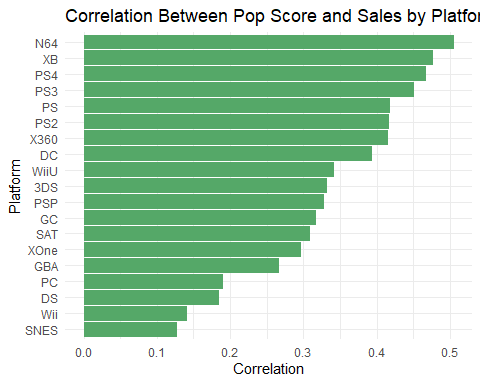
**print**(**head**(cor\_publisher\_pop))

## # A tibble: 6 × 3  
## Publisher n correlation  
## <chr> <int> <dbl>  
## 1 Sony Computer Entertainment 226 0.470  
## 2 Activision 310 0.400  
## 3 Take-Two Interactive 153 0.365  
## 4 Electronic Arts 505 0.297  
## 5 Else 2334 0.288  
## 6 Ubisoft 296 0.199

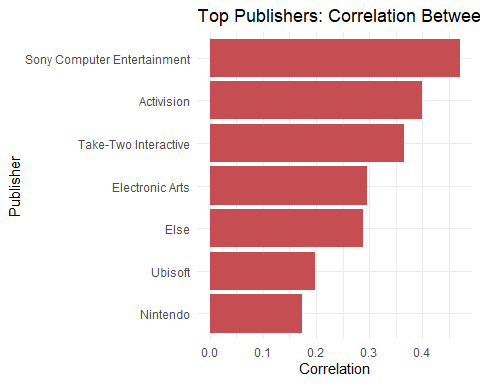
***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Visuals for Correlation Analysis***  
**library**(ggplot2)  
**library**(dplyr)  
  
*# Reuse this function to prepare grouped correlation data*  
compute\_group\_correlation <- **function**(df, group\_var, score\_var) {  
 df **%>%**  
 **group\_by**(.data[[group\_var]]) **%>%**  
 **summarise**(  
 n = **sum**(**!is.na**(.data[[score\_var]]) **&** **!is.na**(Global\_Sales)),  
 correlation = **if** (n **>=** 2) **cor**(.data[[score\_var]], Global\_Sales, use = "complete.obs", method = "pearson") **else** NA\_real\_  
 ) **%>%**  
 **filter**(**!is.na**(correlation)) **%>%**  
 **arrange**(**desc**(correlation))  
}  
  
*# Create data frames*  
cor\_genre\_pop <- **compute\_group\_correlation**(data, "Genre", "Pop")  
cor\_platform\_pop <- **compute\_group\_correlation**(data, "Platform", "Pop")  
cor\_publisher\_pop <- **compute\_group\_correlation**(data, "Publisher", "Pop")  
  
*# Bar Plot: Correlation by Genre*  
**ggplot**(cor\_genre\_pop, **aes**(x = **reorder**(Genre, correlation), y = correlation)) **+**  
 **geom\_bar**(stat = "identity", fill = "#4C72B0") **+**  
 **coord\_flip**() **+**  
 **labs**(title = "Correlation Between Pop Score and Sales by Genre",  
 x = "Genre", y = "Correlation") **+**  
 **theme\_minimal**()



*# Bar Plot: Correlation by Platform*  
**ggplot**(cor\_platform\_pop **%>%** **filter**(n **>=** 10), **aes**(x = **reorder**(Platform, correlation), y = correlation)) **+**  
 **geom\_bar**(stat = "identity", fill = "#55A868") **+**  
 **coord\_flip**() **+**  
 **labs**(title = "Correlation Between Pop Score and Sales by Platform (n ≥ 10)",  
 x = "Platform", y = "Correlation") **+**  
 **theme\_minimal**()



*# Bar Plot: Correlation by Publisher (top 10 only for readability)*  
**ggplot**(cor\_publisher\_pop **%>%** **filter**(n **>=** 10) **%>%** **top\_n**(10, correlation),   
 **aes**(x = **reorder**(Publisher, correlation), y = correlation)) **+**  
 **geom\_bar**(stat = "identity", fill = "#C44E52") **+**  
 **coord\_flip**() **+**  
 **labs**(title = "Top Publishers: Correlation Between Pop Score and Sales",  
 x = "Publisher", y = "Correlation") **+**  
 **theme\_minimal**()



***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## Input Console Column***  
**library**(dplyr)  
*# Step 1: Clean Platform values (remove extra spaces, convert to character)*  
data**$**Platform <- **trimws**(**as.character**(data**$**Platform))  
  
*# Step 2: Define handheld platforms*  
handheld\_platforms <- **c**("GB", "GBA", "DS", "3DS", "PSP")  
  
*# Step 3: Create Console column*  
data**$**Console <- **ifelse**(data**$**Platform **%in%** handheld\_platforms, "H", "S")  
  
**table**(data**$**Console, useNA = "ifany")

##   
## H S   
## 1107 3062

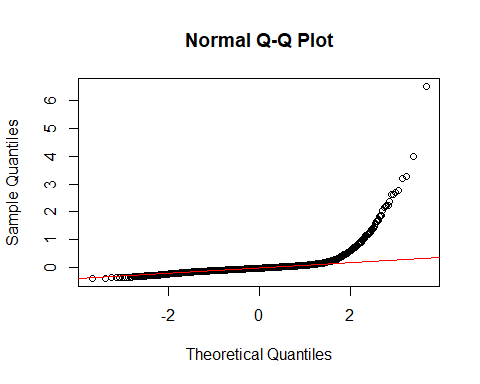
**write.csv**(data, file = "C:/Users/slhor/Downloads/modified\_dataset.csv", row.names = FALSE)  
  
  
***##\_\_\_\_\_\_\_\_\_\_\_***  
***## Regression***  
*# Ensure categorical variables are treated as factors*  
data**$**Genre <- **as.factor**(data**$**Genre)  
data**$**Console <- **as.factor**(data**$**Console)  
data**$**Publisher <- **as.factor**(data**$**Publisher)  
  
*# Fit the linear model*  
model <- **lm**(avg\_annual\_sales **~** Pop **+** Genre **+** Console **+** Publisher, data = data)  
  
*# View the summary*  
**summary**(model)

##   
## Call:  
## lm(formula = avg\_annual\_sales ~ Pop + Genre + Console + Publisher,   
## data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.4043 -0.0968 -0.0369 0.0283 6.5261   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.135620 0.018837 7.200 7.15e-13 \*\*\*  
## Pop 0.088644 0.006686 13.259 < 2e-16 \*\*\*  
## GenreAdventure -0.071292 0.024412 -2.920 0.003516 \*\*   
## GenreFighting -0.046057 0.020160 -2.285 0.022393 \*   
## GenreMisc 0.035666 0.019020 1.875 0.060832 .   
## GenrePlatform -0.024828 0.017414 -1.426 0.154024   
## GenrePuzzle -0.081001 0.027077 -2.991 0.002793 \*\*   
## GenreRacing -0.019511 0.017981 -1.085 0.277947   
## GenreRole-Playing -0.057320 0.014945 -3.835 0.000127 \*\*\*  
## GenreShooter 0.024073 0.015007 1.604 0.108781   
## GenreSimulation -0.014657 0.024275 -0.604 0.546029   
## GenreSports 0.003625 0.015656 0.232 0.816924   
## GenreStrategy -0.116967 0.021891 -5.343 9.63e-08 \*\*\*  
## ConsoleS 0.040067 0.009779 4.097 4.26e-05 \*\*\*  
## PublisherElectronic Arts -0.010063 0.019961 -0.504 0.614210   
## PublisherElse -0.060913 0.016458 -3.701 0.000218 \*\*\*  
## PublisherNintendo 0.174986 0.021390 8.181 3.74e-16 \*\*\*  
## PublisherSony Computer Entertainment -0.015976 0.023559 -0.678 0.497746   
## PublisherTake-Two Interactive 0.024851 0.026853 0.925 0.354805   
## PublisherUbisoft -0.024541 0.021878 -1.122 0.262051   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2655 on 4074 degrees of freedom  
## (75 observations deleted due to missingness)  
## Multiple R-squared: 0.124, Adjusted R-squared: 0.1199   
## F-statistic: 30.35 on 19 and 4074 DF, p-value: < 2.2e-16

*# Fit the linear model*  
model <- **lm**(avg\_annual\_sales **~** Pop **+** Genre **+** Console **+** Publisher, data = data)  
  
*# View the summary*  
**summary**(model)

##   
## Call:  
## lm(formula = avg\_annual\_sales ~ Pop + Genre + Console + Publisher,   
## data = data)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.4043 -0.0968 -0.0369 0.0283 6.5261   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.135620 0.018837 7.200 7.15e-13 \*\*\*  
## Pop 0.088644 0.006686 13.259 < 2e-16 \*\*\*  
## GenreAdventure -0.071292 0.024412 -2.920 0.003516 \*\*   
## GenreFighting -0.046057 0.020160 -2.285 0.022393 \*   
## GenreMisc 0.035666 0.019020 1.875 0.060832 .   
## GenrePlatform -0.024828 0.017414 -1.426 0.154024   
## GenrePuzzle -0.081001 0.027077 -2.991 0.002793 \*\*   
## GenreRacing -0.019511 0.017981 -1.085 0.277947   
## GenreRole-Playing -0.057320 0.014945 -3.835 0.000127 \*\*\*  
## GenreShooter 0.024073 0.015007 1.604 0.108781   
## GenreSimulation -0.014657 0.024275 -0.604 0.546029   
## GenreSports 0.003625 0.015656 0.232 0.816924   
## GenreStrategy -0.116967 0.021891 -5.343 9.63e-08 \*\*\*  
## ConsoleS 0.040067 0.009779 4.097 4.26e-05 \*\*\*  
## PublisherElectronic Arts -0.010063 0.019961 -0.504 0.614210   
## PublisherElse -0.060913 0.016458 -3.701 0.000218 \*\*\*  
## PublisherNintendo 0.174986 0.021390 8.181 3.74e-16 \*\*\*  
## PublisherSony Computer Entertainment -0.015976 0.023559 -0.678 0.497746   
## PublisherTake-Two Interactive 0.024851 0.026853 0.925 0.354805   
## PublisherUbisoft -0.024541 0.021878 -1.122 0.262051   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2655 on 4074 degrees of freedom  
## (75 observations deleted due to missingness)  
## Multiple R-squared: 0.124, Adjusted R-squared: 0.1199   
## F-statistic: 30.35 on 19 and 4074 DF, p-value: < 2.2e-16

***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***## QQ-Plot of Residuals***  
**qqnorm**(model**$**residuals)  
**qqline**(model**$**residuals, col = "red")



***##\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***  
***##Console Scatter Plot***  
**library**(ggplot2)  
  
**ggplot**(data, **aes**(x = Pop, y = avg\_annual\_sales)) **+**  
 **geom\_point**(**aes**(color = Console)) **+**  
 **facet\_wrap**(**~** Console) **+**  
 **labs**(title = "Pop Score vs Avg Global Sales by Console",  
 x = "Pop Score", y = "Avg Global Sales") **+**  
 **theme\_minimal**()

## Warning: Removed 75 rows containing missing values or values outside the scale range  
## (`geom\_point()`).

