as4_6607

2024-11-22

```
# Abdallah Chidjou
# Citation: (source of help: Lecture note, googling in general, stackoverflow, and chatgpt)
```

Question 1

```
import pandas as pd

# Load the dataset
file_path = 'Sales_Data.csv'
sales_data = pd.read_csv(file_path)

# a. Ensure that the 'Date' column is converted to datetime format
if 'Date' in sales_data.columns:
    sales_data['Date'] = pd.to_datetime(sales_data['Date'], errors='coerce')

# Display the first few rows of the dataset to confirm changes
sales_data.head()
```

```
##         Date Region    Product  Units Sold      Revenue
## 0 2022-01-31   East  Product A         478  6946.477740
## 1 2022-02-28   West  Product A         811  1862.496458
## 2 2022-03-31  North  Product B         618  2658.223424
## 3 2022-04-30  North  Product B         832  5296.000964
## 4 2022-05-31  South  Product A         873  8427.695285
```

```
# b. Add a new column 'Profit' based on the region
sales_data['Profit'] = sales_data['Revenue'] * sales_data['Region'].map({
    'North': 0.3,
    'South': 0.4
}).fillna(0.25)

# Display the updated dataset with the new 'Profit' column
print(sales_data.head())
```

```
##         Date Region    Product  Units Sold      Revenue       Profit
## 0 2022-01-31   East  Product A         478  6946.477740  1736.619435
## 1 2022-02-28   West  Product A         811  1862.496458   465.624114
## 2 2022-03-31  North  Product B         618  2658.223424   797.467027
## 3 2022-04-30  North  Product B         832  5296.000964  1588.800289
## 4 2022-05-31  South  Product A         873  8427.695285  3371.078114
```

```
# c. Calculate total units sold and average profit per unit sold for each product
summary_data = sales_data.groupby('Product').agg(
    total_units_sold=('Units Sold', 'sum'),
    average_profit_per_unit=
    ('Profit', lambda x: x.sum() / sales_data.loc[x.index, 'Units Sold'].sum())).reset_index()

# Display the new summary DataFrame
print(summary_data)
```

```
##      Product  total_units_sold  average_profit_per_unit
## 0  Product A              9581                 3.238473
## 1  Product B             11412                 2.693498
## 2  Product C              2826                 6.121559
```

```
# d. Filter the original dataset to include only rows where:
# The Date falls in the year 2023 and Revenue is above
# the median revenue for all rows
median_revenue = sales_data['Revenue'].median()
filtered_data = sales_data[(sales_data['Date'].dt.year == 2023) &
(sales_data['Revenue'] > median_revenue)]

# Display the filtered DataFrame
print(filtered_data)
```

```
##           Date Region    Product  Units Sold      Revenue       Profit
## 15 2023-04-30  North  Product A         444  5816.125435  1744.837630
## 16 2023-05-31  South  Product B         801  7795.184313  3118.073725
## 19 2023-08-31  South  Product B         119  6742.796248  2697.118499
## 22 2023-11-30  North  Product B         327  5305.322969  1591.596891
## 23 2023-12-31   East  Product A         196  7084.145492  1771.036373
```

```
# e. For the filtered dataset: Group by Region and
# calculate total Profit and Units Sold for each region
region_summary = filtered_data.groupby('Region').agg(
    total_profit=('Profit', 'sum'),
    total_units_sold=('Units Sold', 'sum')
).sort_values(by='total_profit', ascending=False).reset_index()

# Display the grouped and sorted DataFrame
print(region_summary)
```

```
##   Region  total_profit  total_units_sold
## 0  South   5815.192224               920
## 1  North   3336.434521               771
## 2   East   1771.036373               196
```

```
# Export the grouped data to a new CSV file called 'region_summary.csv'
region_summary.to_csv('region_summary.csv', index=False)
```

Question 2

```
library(dplyr)
```

```
##
## Attaching package: ’dplyr’

## The following objects are masked from ’package:stats’:
##
##     filter, lag

## The following objects are masked from ’package:base’:
##
##     intersect, setdiff, setequal, union
```

```
library(tidyr)
```

```
# Read in the CSV file
df <- read.csv("messy_sales_data.csv")
head(df)
```

```
##     Month Product.Region Sales
## 1 2023-01        A.North   500
## 2 2023-01        A.South   400
## 3 2023-01        B.North   600
## 4 2023-02        A.North   550
## 5 2023-02        B.South   700
```

```
# a. Separate the Product.Region column into Product and Region
df <- df %>%
  separate(Product.Region, into = c("Product", "Region"), sep = "\\.")
print(df)
```

```
##     Month Product Region Sales
## 1 2023-01       A  North   500
## 2 2023-01       A  South   400
## 3 2023-01       B  North   600
## 4 2023-02       A  North   550
## 5 2023-02       B  South   700
```

```
# b. Transform the dataset so that each Product becomes a column
# Summarize sales by Month and Region
df_transform <- df %>%
  pivot_wider(names_from = Product, values_from = Sales, values_fill = 0) #list(Sales = 0)
print(df_transform)
```

```
## # A tibble: 4 x 4
##   Month   Region     A     B
##   <chr>   <chr>  <int> <int>
## 1 2023-01 North    500   600
## 2 2023-01 South    400     0
## 3 2023-02 North    550     0
## 4 2023-02 South      0   700
```

```
# c. Calculate an additional column for total sales across all products
df_transform <- df_transform %>%
   mutate(Total_Sales = rowSums(select(., -c(Month, Region))))
print(df_transform)
```

```
## # A tibble: 4 x 5
##   Month   Region     A     B Total_Sales
##   <chr>   <chr>  <int> <int>       <dbl>
## 1 2023-01 North    500   600        1100
## 2 2023-01 South    400     0         400
## 3 2023-02 North    550     0         550
## 4 2023-02 South      0   700         700
```

```
# d. Filter rows where total sales exceed the average total sales
average_total_sales <- mean(df_transform$Total_Sales)
df_filtered <- df_transform %>%
  filter(Total_Sales > average_total_sales)
print(df_filtered)
```

```
## # A tibble: 2 x 5
##   Month   Region     A     B Total_Sales
##   <chr>   <chr>  <int> <int>       <dbl>
## 1 2023-01 North    500   600        1100
## 2 2023-02 South      0   700         700
```

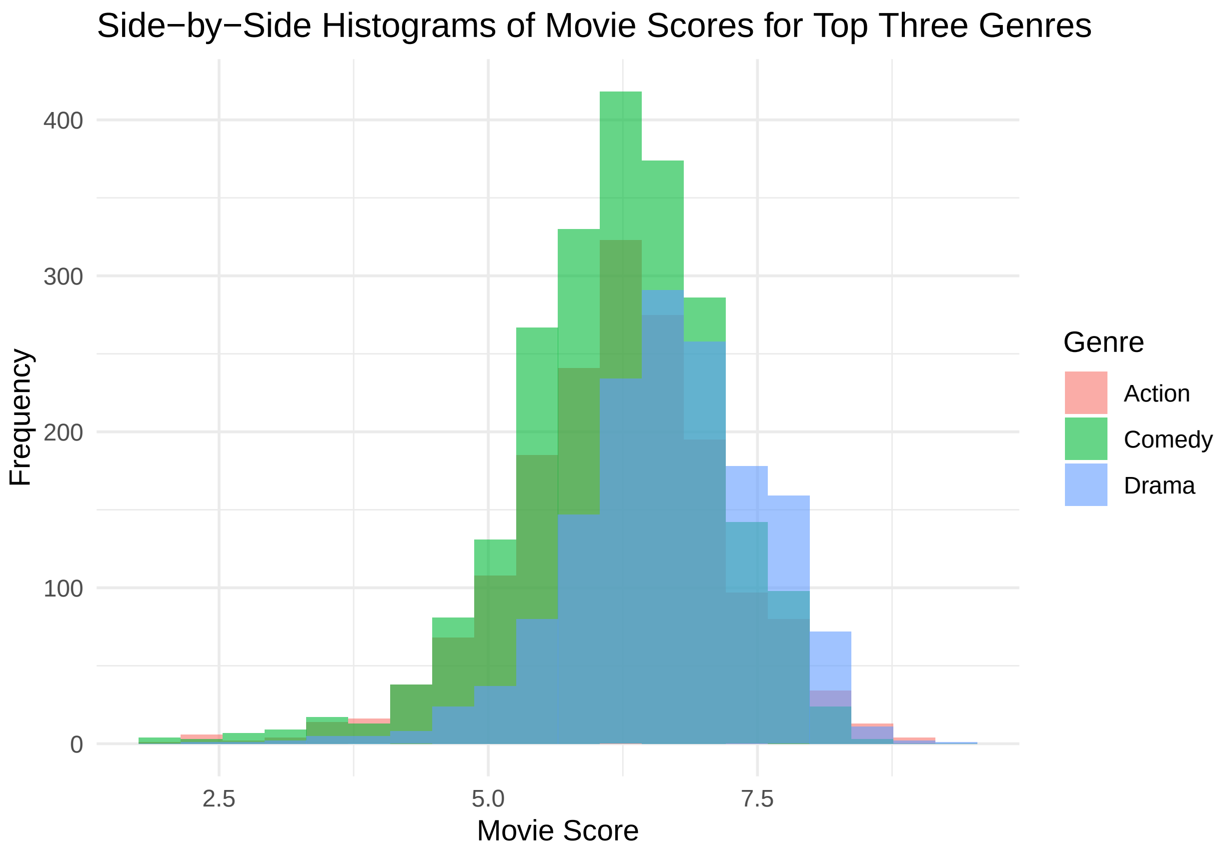
```
# Step e: Sort the resulting dataset by Month and Region
df_sorted <- df_filtered %>%
  arrange(Month, Region)

# Write the tidy dataset to a CSV file
write.csv(df_sorted, "tidy_sales_data.csv", row.names = FALSE)
```

Question 3

```
# Load the required libraries
library(ggplot2)
library(dplyr)

# Load the dataset from the URL
url <- "https://raw.githubusercontent.com/Juanets/movie-stats/master/movies.csv"
movies <- read.csv(url)
#
# Determine the top three genres
top_genres <- movies %>%
  group_by(genre) %>%
```

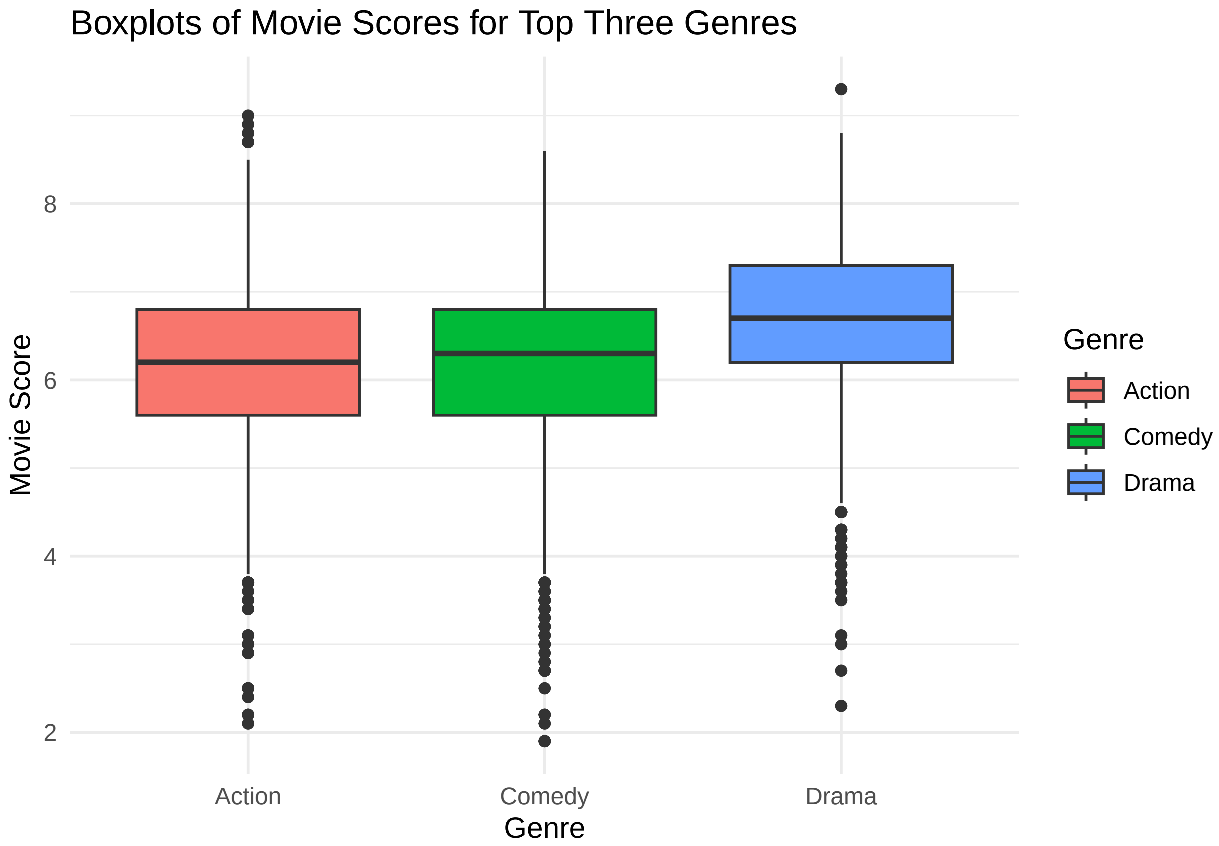
```r
  summarise(count = n()) %>%
  arrange(desc(count)) %>%
  head(3) %>%
  pull(genre)

# Filter the dataset to only include the top three genres
filtered_movies <- movies %>% filter(genre %in% top_genres)

# a. Plot side-by-side histograms of movie scores for the top three genres
ggplot(filtered_movies, aes(x = score, fill = genre)) +
  geom_histogram(alpha = 0.6, position = "identity", bins = 20) +
  labs(title = "Side-by-Side Histograms of Movie Scores for Top Three Genres",
       x = "Movie Score", y = "Frequency") +
  scale_fill_discrete(name = "Genre") +
  theme_minimal()
```

```
## Warning: Removed 3 rows containing non-finite outside the scale range
## (‘stat_bin()‘).
```



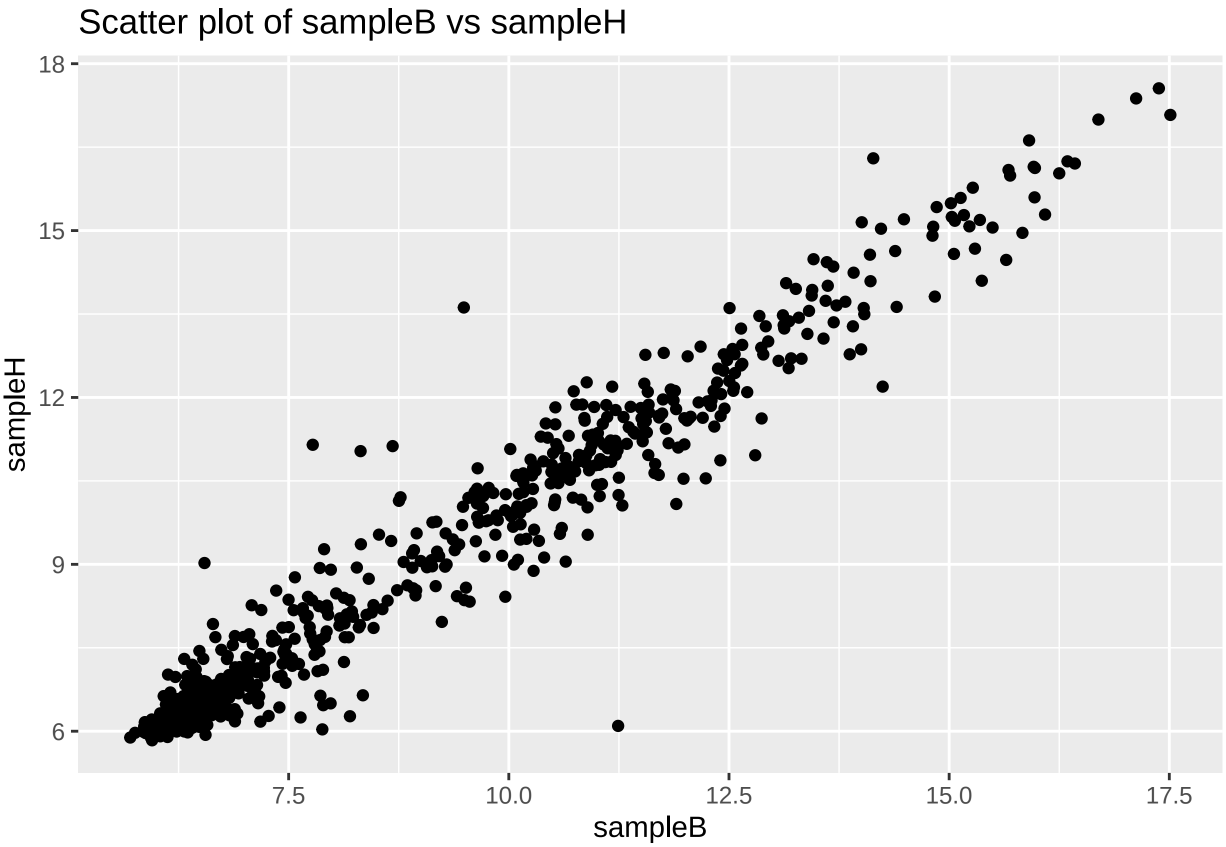
```r
# b. Plot side-by-side boxplots of movie scores for the top three genres
ggplot(filtered_movies, aes(x = genre, y = score, fill = genre)) +
  geom_boxplot() +
  labs(title = "Boxplots of Movie Scores for Top Three Genres",
       x = "Genre", y = "Movie Score") +
  scale_fill_discrete(name = "Genre") +
  theme_minimal()
```

```
## Warning: Removed 3 rows containing non-finite outside the scale range
## (`stat_boxplot()`).
```

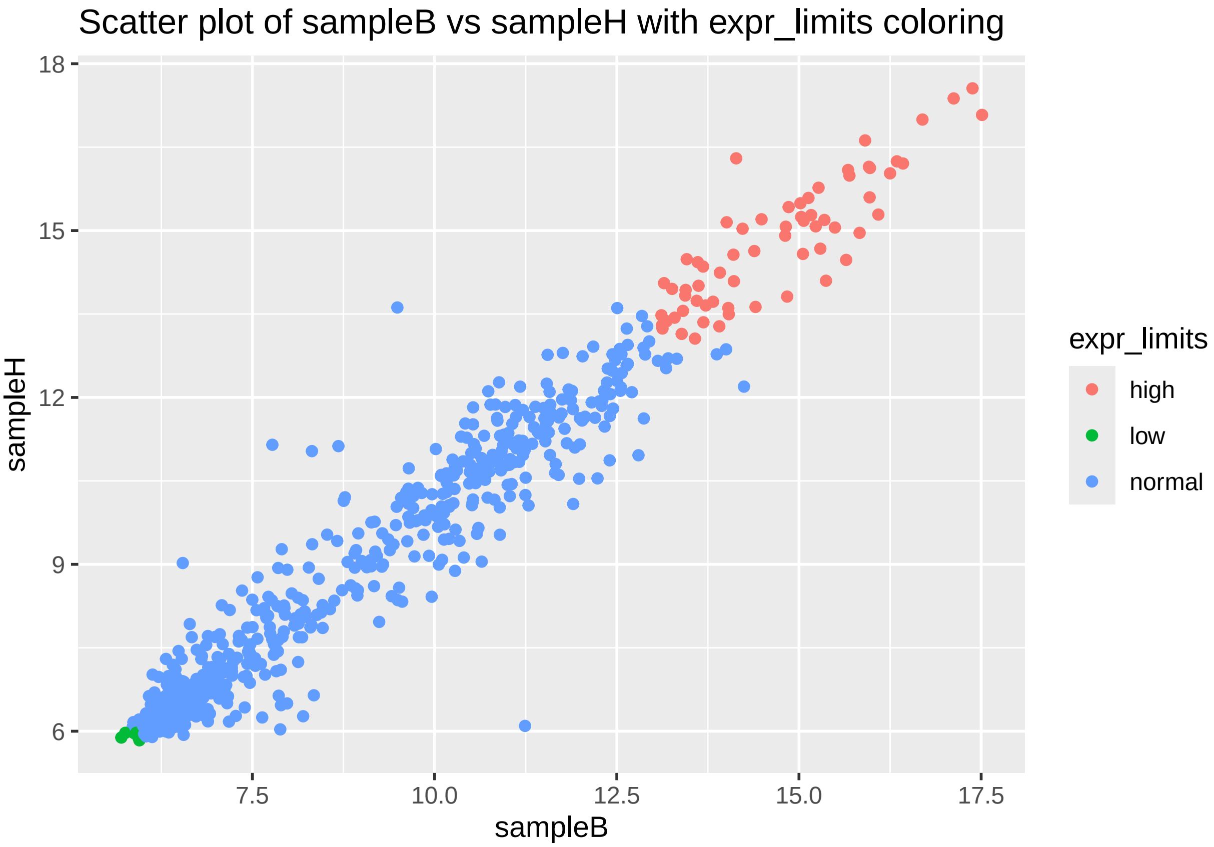


Question 4

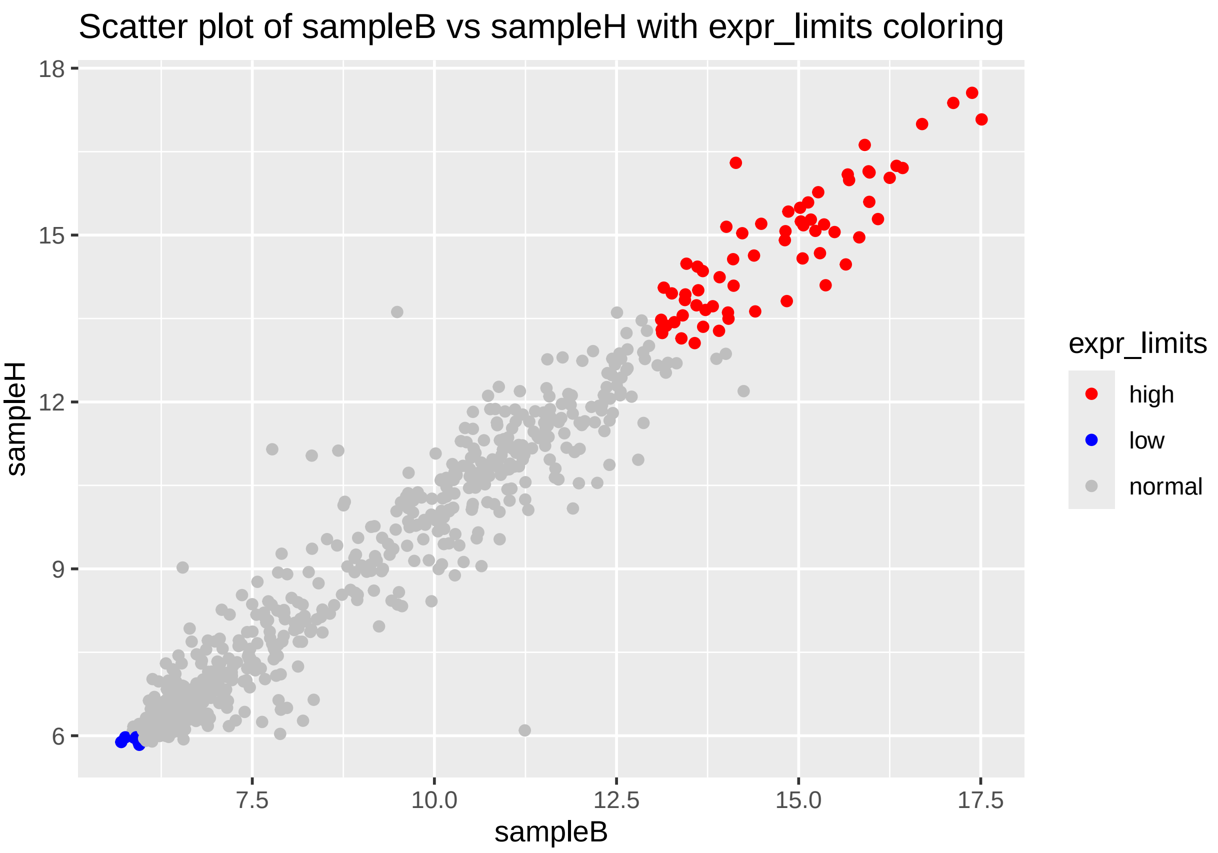
```
# 1. Read the CSV file into an object called intenseData
intenseData <- read.csv("ex12_normalized_intensities.csv")

# 2. Create a scatter plot with ggplot2
library(ggplot2)

# Creating a simple scatter plot of sampleB on the x-axis and sampleH on the y-axis
p <- ggplot(intenseData, aes(x = sampleB, y = sampleH)) +
  geom_point() +
  labs(title = "Scatter plot of sampleB vs sampleH",
       x = "sampleB",
       y = "sampleH")

# Display the initial scatter plot
print(p)
```



```
# 3. Add a new column called expr_limits
intenseData$expr_limits <- ifelse(intenseData$sampleB > 13 & intenseData$sampleH > 13,
                                  "high",
                                  ifelse(intenseData$sampleB < 6 & intenseData$sampleH < 6,
                                         "low", "normal"))

# 4. Modify the plot to color points by expr_limits
p <- ggplot(intenseData, aes(x = sampleB, y = sampleH, color = expr_limits)) +
  geom_point() +
  labs(title = "Scatter plot of sampleB vs sampleH with expr_limits coloring",
       x = "sampleB",
       y = "sampleH")

# Display the modified scatter plot
print(p)
```



```
# 5. Add a layer to change point colors to blue, grey, and red
p2 <- p + scale_color_manual(values = c("low" = "blue", "normal" = "grey", "high" = "red"))

# Display the final scatter plot
print(p2)
```



```
# 6. Save the final plot to a PDF file
ggsave("scatter_plot_expr_limits.pdf", plot = p2)
```

```
## Saving 6.5 x 4.5 in image
```