Tech Layoff

Hilena Shekour

```
layoff <- read.csv(("layoffs_data.csv"))
layoffs_clean <- na.omit(layoff)
layoffs_us <- layoffs_clean[layoffs_clean$Country == "United States", ]

# Convert the Date column to Date format
layoffs_us$Date <- as.Date(layoffs_us$Date, format="%Y-%m-%d")

# Extract year and month from the Date
layoffs_us$Year <- format(layoffs_us$Date, "%Y")
layoffs_us$Month <- format(layoffs_us$Date, "%m")

# Group by Year and Month, then calculate the average layoffs per group
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
```

```
layoffs_by_month <- layoffs_us %>%
    group_by(Year, Month) %>%
    mutate(Low_Cutoff = quantile(Laid_Off_Count, 0.25, na.rm = TRUE),
           High_Cutoff = quantile(Laid_Off_Count, 0.75, na.rm = TRUE),
           Layoff_Category = case_when(
               Laid_Off_Count <= Low_Cutoff ~ "Low Layoffs",
               Laid_Off_Count > Low_Cutoff & Laid_Off_Count <= High_Cutoff ~ "Medium Layoffs",
               Laid_Off_Count > High_Cutoff ~ "High Layoffs"
           ))

stage_mapping <- list(
    "Seed" = "Early Stage",
    "Series A" = "Early Stage",
    "Series B" = "Mid Stage",
    "Series C" = "Mid Stage",
    "Series D" = "Mid Stage",
    "Series E" = "Late Stage",
    "Series F" = "Late Stage",
    "Series G" = "Late Stage",
    "Series H" = "Late Stage",
    "Series I" = "Late Stage",
```

```
    "Post-IPO" = "Established",
    "Acquired" = "Established",
    "Private Equity" = "Established",
    "Subsidiary" = "Established",
    "Unknown" = "Unknown"
)

# Create a new column with the recategorized stages
layoffs_by_month$Stage_Categorized <- unlist(stage_mapping[layoffs_by_month$Stage])

# List of U.S. cities (excluding international cities)
us_cities <- c("Denver", "SF Bay Area", "New York City", "Austin", "Los Angeles",
               "Lehi", "Seattle", "Pittsburgh", "Phoenix", "Detroit", "Portland",
               "Boston", "San Diego", "Boulder", "Miami", "Atlanta", "Corvallis",
               "Norwalk", "Milwaukee", "Kansas City", "Salt Lake City", "Raleigh",
               "Chicago", "Santa Barbara", "Tampa Bay", "Washington D.C.",
               "Baltimore", "Dallas", "Cincinnati", "Philadelphia", "Nashua",
               "Madison", "Minneapolis", "Boise", "Columbus", "Oxford",
               "Wilmington", "Nashville", "Reno", "Burlington", "Sacramento",
               "Dover", "Logan", "Nebraska City", "San Luis Obispo",
               "Indianapolis", "Stamford", "Spokane", "Bend", "Las Vegas",
               "Santa Fe", "Missoula", "Ann Arbor")

# Filter the dataset to keep only U.S. cities
layoffs_by_month <- layoffs_by_month[layoffs_by_month$Location_HQ %in% us_cities, ]

# List unique values
unique(layoffs_by_month$Region)
```

```
## Warning: Unknown or uninitialised column: `Region`.
```

```
## NULL
```

```
west_cities <- c("Denver", "SF Bay Area", "Los Angeles", "Lehi", "Seattle", "Phoenix",
                 "Portland", "San Diego", "Boulder", "Salt Lake City", "Santa Barbara",
                 "Reno", "Sacramento", "San Luis Obispo", "Spokane", "Bend", "Las Vegas",
                 "Logan")
northeast_cities <- c("New York City", "Pittsburgh", "Boston", "Norwalk", "Philadelphia",
                      "Nashua", "Burlington")
south_cities <- c("Austin", "Miami", "Atlanta", "Corvallis", "Raleigh", "Tampa Bay",
                  "Washington D.C.", "Baltimore", "Dallas", "Nashville", "Santa Fe",
                  "Stamford")
midwest_cities <- c("Detroit", "Milwaukee", "Kansas City", "Chicago", "Cincinnati",
                    "Madison", "Minneapolis", "Boise", "Columbus", "Oxford",
                    "Wilmington", "Indianapolis", "Nebraska City", "Missoula",
                    "Ann Arbor", "Dover")


# Update city_to_region mapping
city_to_region <- list(
  "West" = west_cities,
  "Northeast" = northeast_cities,
  "South" = south_cities,
  "Midwest" = midwest_cities
)
```

```r
# Function to map cities to regions
get_region <- function(city) {
  for (region in names(city_to_region)) {
    if (city %in% city_to_region[[region]]) {
      return(region)
    }
  }
  return(NA)
}
# Reapply the mapping
layoffs_by_month$Region <- sapply(layoffs_by_month$Location_HQ, get_region)

# Check the distribution of regions and NA values
table(layoffs_by_month$Region)
```

```
##
##   Midwest Northeast     South      West
##        60       244        60       598
```

```r
# Generate a grid of unique combinations of cities and stages
unique_cities <- unique(layoffs_by_month$Location_HQ)
grid <- expand.grid(
  Stage_Categorized = unique(layoffs_by_month$Stage_Categorized),
  Location_HQ = unique_cities
)
# Generate a grid of unique combinations of stages and regions
grid <- expand.grid(
  Stage_Categorized = unique(layoffs_by_month$Stage_Categorized),
  Region = unique(layoffs_by_month$Region)
)
```

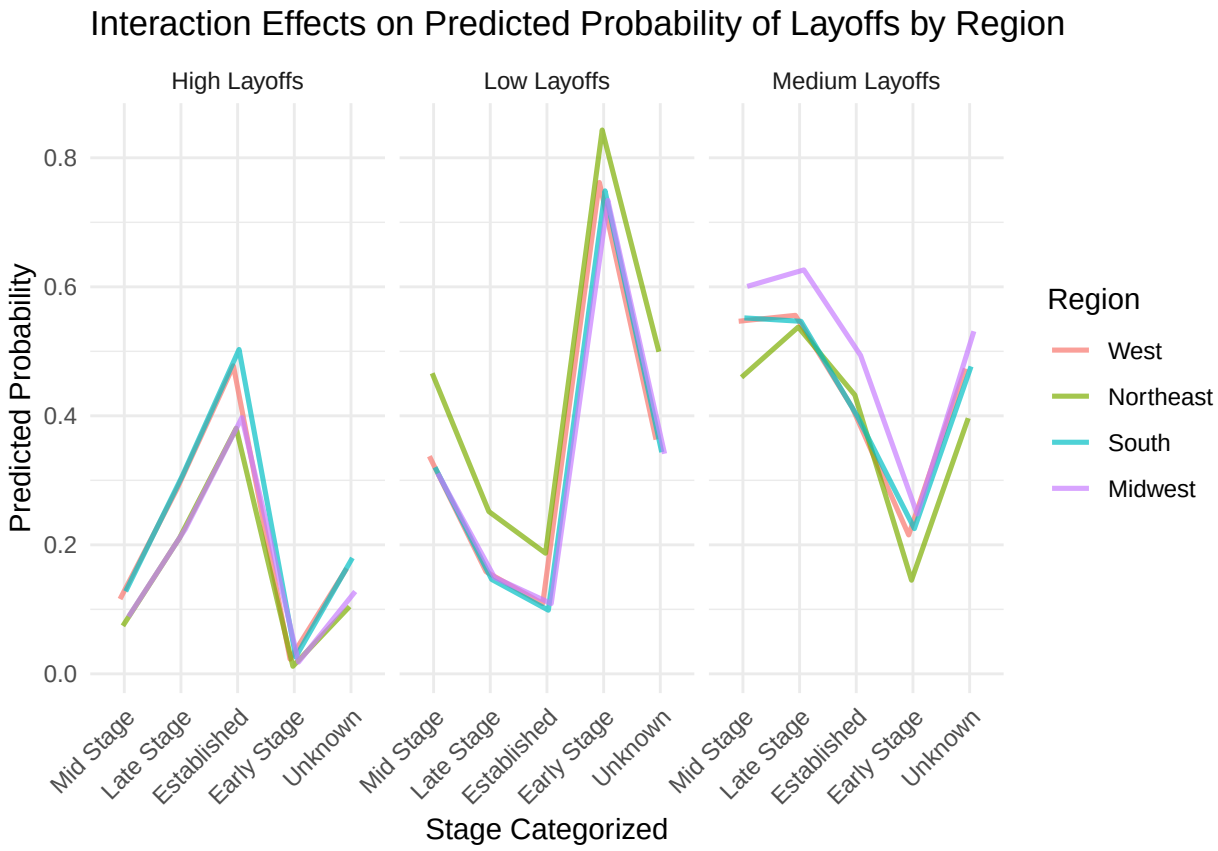
```r
mlr_model <- multinom(Layoff_Category ~ Stage_Categorized + Region, data = layoffs_by_month)
```

```
## # weights:  27 (16 variable)
## initial  value 1056.865022
## iter  10 value 921.131944
## iter  20 value 909.235804
## final  value 909.098951
## converged
```

```r
predictions <- predict(mlr_model, newdata = layoffs_by_month)

# Ensure actual labels are factors
layoffs_by_month$Layoff_Category <- factor(layoffs_by_month$Layoff_Category)

# Ensure predicted labels are factors
predictions <- factor(predictions, levels = levels(layoffs_by_month$Layoff_Category))



# Predict probabilities using the region data
predictions <- predict(mlr_model, newdata = grid, type = "probs")

# Combine the predictions back into the grid
grid <- cbind(grid, predictions)
```

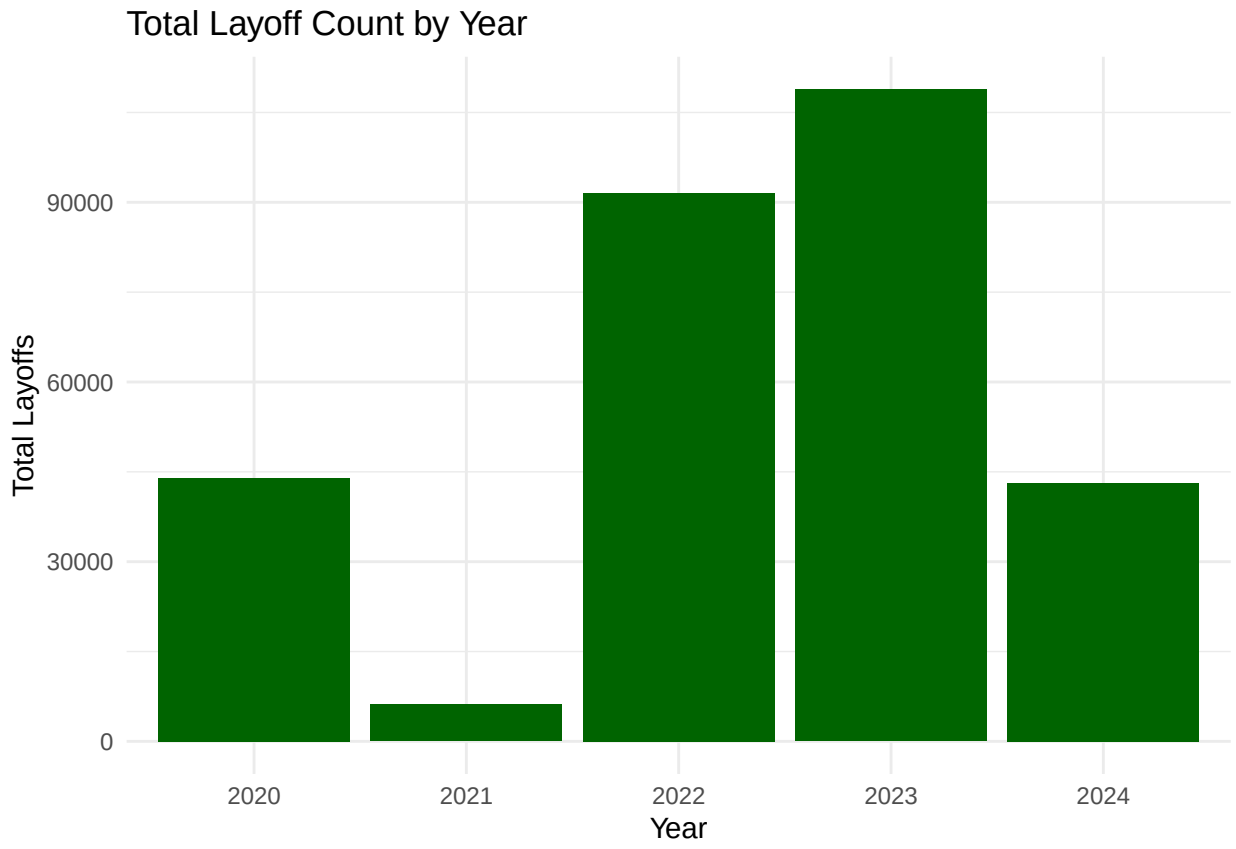
```r
# Convert to long format for plotting
library(tidyr)
long_grid <- gather(grid, Layoff_Category, Probability, -Stage_Categorized, -Region)

# Plot interaction effects using geom_line and the Region column
library(ggplot2)
ggplot(long_grid, aes(x = Stage_Categorized, y = Probability, color = Region, group = Region)) +
  geom_line(size = 0.9, position = position_dodge(width = 0.2), alpha = 0.7) +
  facet_wrap(~ Layoff_Category) +
  theme_minimal() +
  labs(title = "Interaction Effects on Predicted Probability of Layoffs by Region", x = "Stage Categori
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## 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.
```

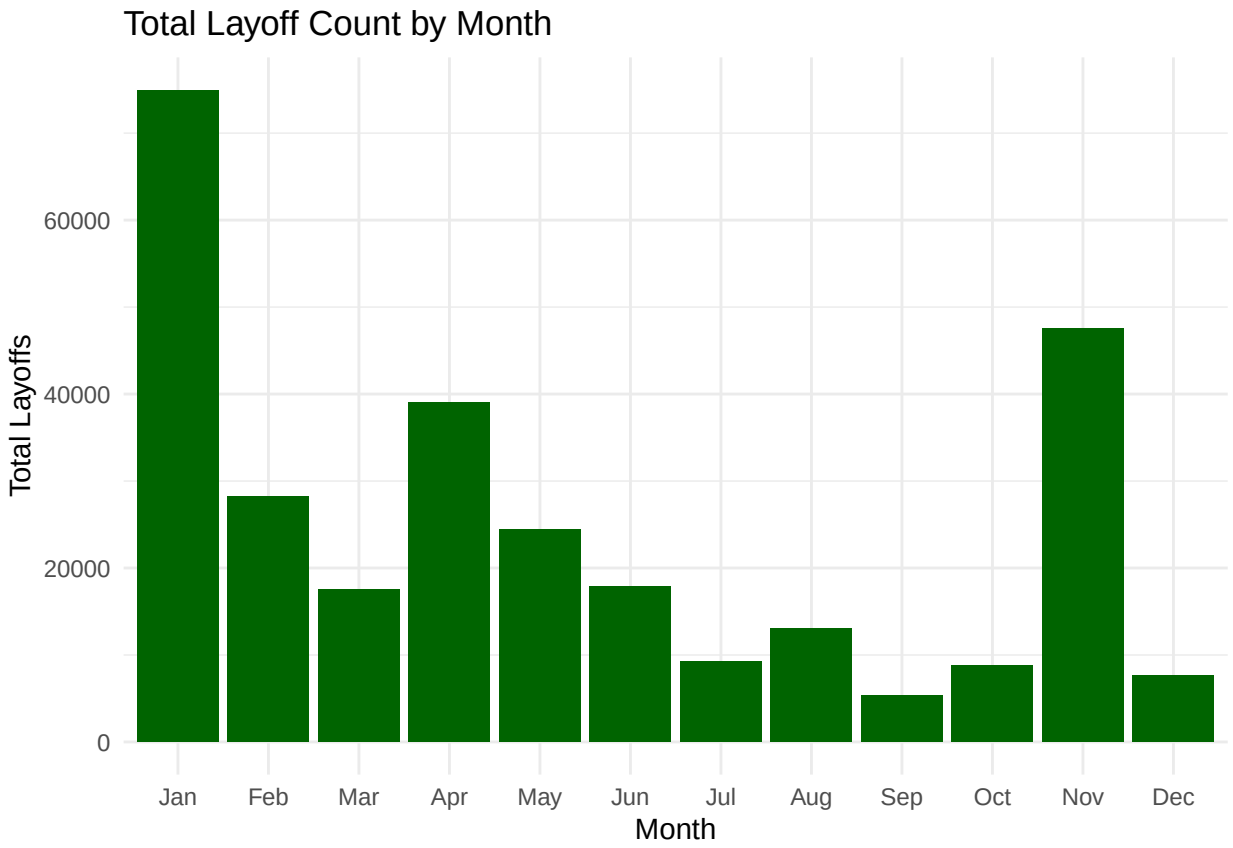



```r
# Layoff count by year
layoffs_by_year <- layoffs_by_month %>%
  group_by(Year) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count))

plot_layoffs_by_year <- ggplot(layoffs_by_year, aes(x = as.factor(Year), y = Total_Layoffs)) +
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Total Layoff Count by Year", x = "Year", y = "Total Layoffs") +
```

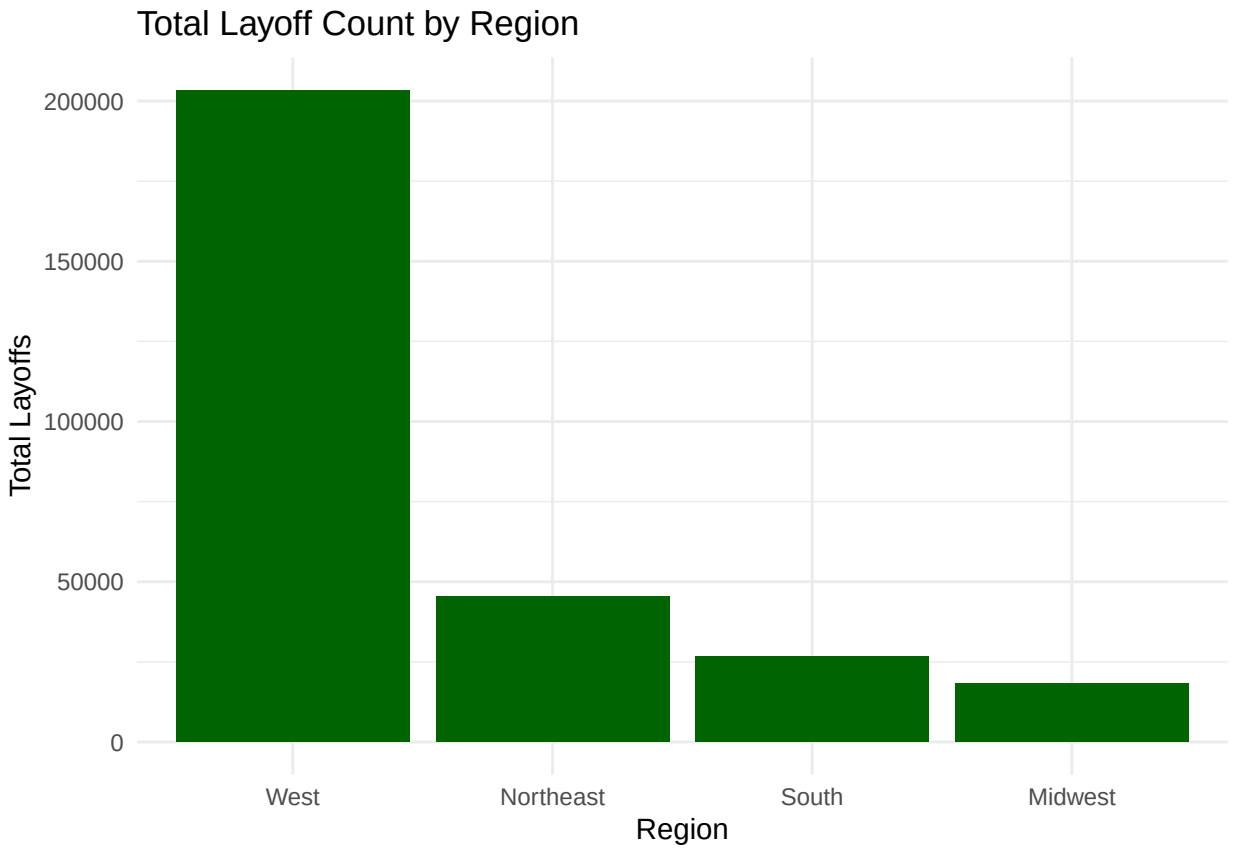
```
  theme_minimal()
plot_layoffs_by_year
```



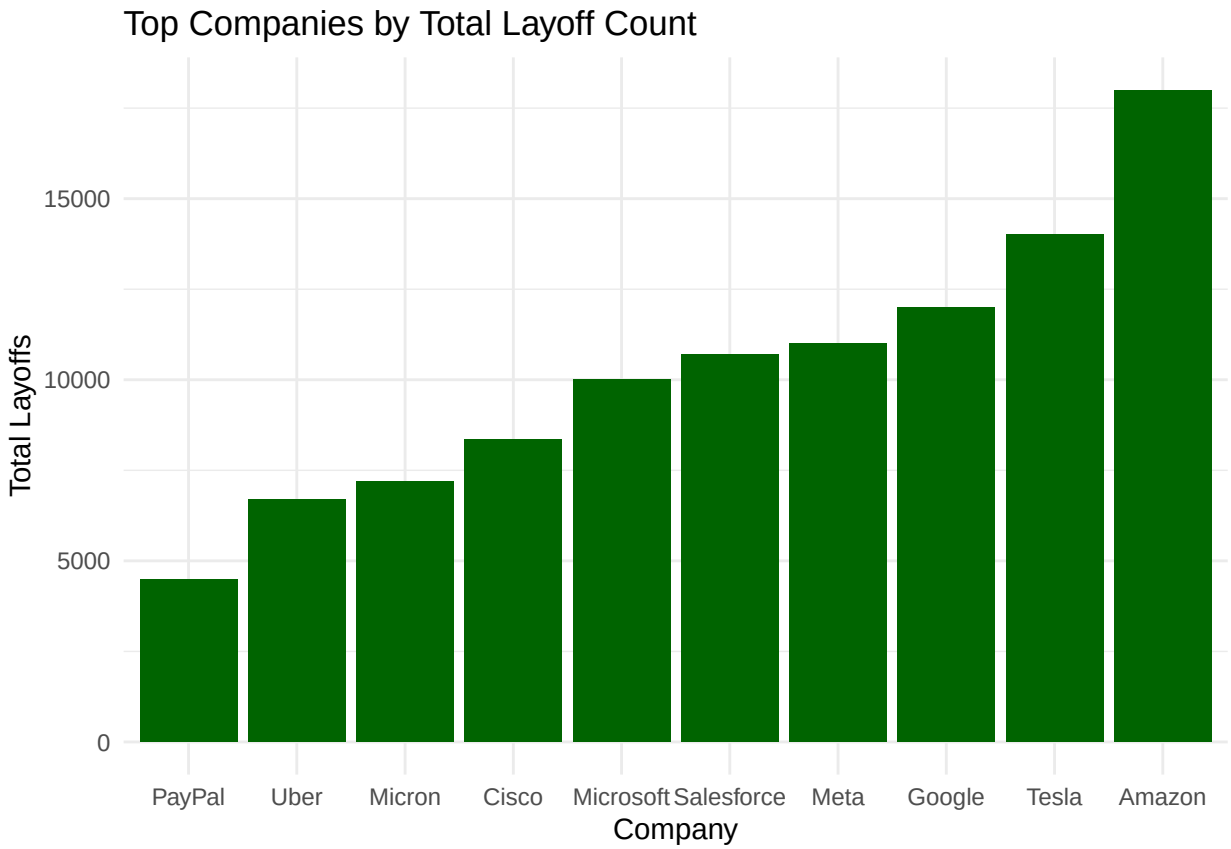

```
# Layoff count by month
layoffs_by_months <- layoffs_by_month %>%
  group_by(Month) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count))

plot_layoffs_by_month <- ggplot(layoffs_by_months, aes(x = as.factor(Month), y = Total_Layoffs)) +
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Total Layoff Count by Month", x = "Month", y = "Total Layoffs") +
  theme_minimal() +
  scale_x_discrete(labels = c("01" = "Jan", "02" = "Feb", "03" = "Mar", "04" = "Apr",
                              "05" = "May", "06" = "Jun", "07" = "Jul", "08" = "Aug",
                              "09" = "Sep", "10" = "Oct", "11" = "Nov", "12" = "Dec"))
plot_layoffs_by_month
```




```r
# Layoff count by state
layoffs_by_state <- layoffs_by_month %>%
  group_by(Region) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count))

plot_layoffs_by_state <- ggplot(layoffs_by_state, aes(x = reorder(Region, -Total_Layoffs), y = Total_Lay
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Total Layoff Count by Region", x = "Region", y = "Total Layoffs") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme_minimal()
plot_layoffs_by_state
```




```r
# Layoff count by company
layoffs_by_company <- layoffs_by_month %>%
  group_by(Company) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count)) %>%
  arrange(desc(Total_Layoffs))  # Sort in descending order of total layoffs

# Select top companies
top_companies <- head(layoffs_by_company, 10)  # Top 10 companies

plot_top_layoffs_by_company <- ggplot(top_companies, aes(x = reorder(Company, Total_Layoffs), y = Total_
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Top Companies by Total Layoff Count", x = "Company", y = "Total Layoffs") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme_minimal()
plot_top_layoffs_by_company
```



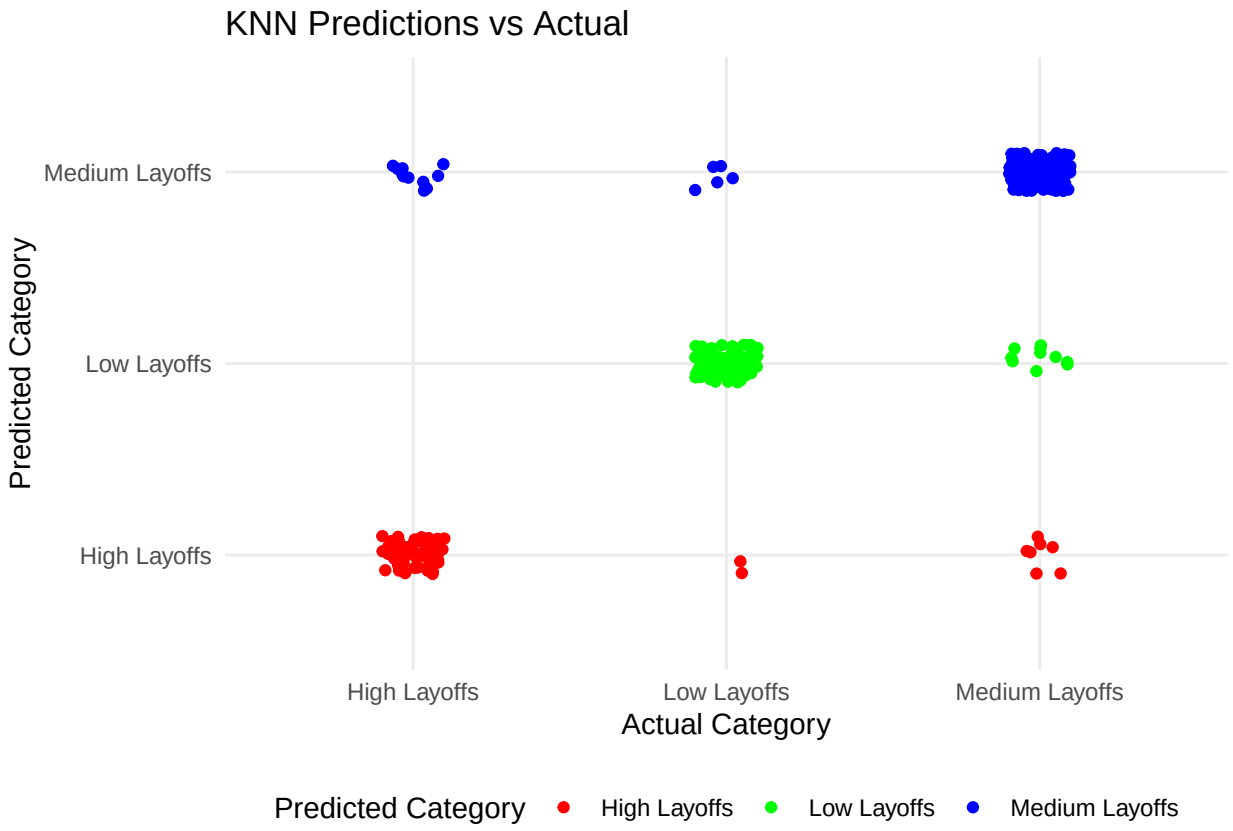

```r
layoffs_by_month$Month <- as.numeric(layoffs_by_month$Month)

layoffs_by_month$Layoff_Category <- factor(layoffs_by_month$Layoff_Category)
numeric_features <- layoffs_by_month[, c("Laid_Off_Count", "Month")]
target <- layoffs_by_month$Layoff_Category
scaled_data <- scale(numeric_features)

# Create a split index
set.seed(42)
index <- sample(1:nrow(layoffs_by_month), size = 0.7 * nrow(layoffs_by_month))

# Create training and testing sets
train_data <- scaled_data[index, ]
train_target <- target[index]
test_data <- scaled_data[-index, ]
test_target <- target[-index]

# Apply KNN, choose an appropriate value for k
k <- 5  # Adjust based on your data
predicted_target <- knn(train = train_data, test = test_data, cl = train_target, k = k)
# Measure the accuracy of the predictions
accuracy <- sum(predicted_target == test_target) / length(test_target)
cat("Accuracy:", accuracy)
```

```
## Accuracy: 0.8788927
```

```r
summary(accuracy)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.8789  0.8789  0.8789  0.8789  0.8789  0.8789
```

```r
# Create a data frame for plotting
plot_data <- data.frame(Actual = test_target, Predicted = predicted_target)

# Scatter plot to show the results
ggplot(plot_data, aes(x = Actual, y = Predicted, color = Predicted)) +
  geom_jitter(width = 0.1, height = 0.1) +
  labs(title = "KNN Predictions vs Actual", x = "Actual Category", y = "Predicted Category", color = "Pr
  theme_minimal() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("red", "green", "blue"))  # Adjust colors for each category
```




```r
# Example accuracies, replace with actual values
knn_accuracy <- 0.8788927  # Example accuracy for KNN
mlr_accuracy <- 0.5218   # Example accuracy for Multinomial Logistic Regression

# Create a data frame
results_df <- data.frame(
  Model = c("KNN", "MLR"),
  Accuracy = c(knn_accuracy, mlr_accuracy)
)
# Generate table
accuracy_table <- kable(results_df, format = "html", col.names = c("Model", "Accuracy"), align = c('l',
```

```r
# Print table
print(accuracy_table)
```

```
## <table>
##  <thead>
##   <tr>
##    <th style="text-align:left;"> Model </th>
##    <th style="text-align:center;"> Accuracy </th>
##   </tr>
##  </thead>
## <tbody>
##   <tr>
##    <td style="text-align:left;"> KNN </td>
##    <td style="text-align:center;"> 0.8788927 </td>
##   </tr>
##   <tr>
##    <td style="text-align:left;"> MLR </td>
##    <td style="text-align:center;"> 0.5218000 </td>
##   </tr>
## </tbody>
## </table>
```

Appendix

```r
# Insert packages you need here
# Set include=FALSE to hide code and output
# Do this when loading packages
library(knitr)
library(ggplot2)
#install.packages("nnet")  # If not already installed
library(nnet)
#install.packages("caret")
library(caret)
#install.packages("plotly")
library(plotly)

library(corrplot)
library(class)
library(GGally)
layoff <- read.csv(("layoffs_data.csv"))
layoffs_clean <- na.omit(layoff)
layoffs_us <- layoffs_clean[layoffs_clean$Country == "United States", ]

# Convert the Date column to Date format
layoffs_us$Date <- as.Date(layoffs_us$Date, format="%Y-%m-%d")

# Extract year and month from the Date
layoffs_us$Year <- format(layoffs_us$Date, "%Y")
layoffs_us$Month <- format(layoffs_us$Date, "%m")

# Group by Year and Month, then calculate the average layoffs per group
library(dplyr)
layoffs_by_month <- layoffs_us %>%
```

```r
    group_by(Year, Month) %>%
    mutate(Low_Cutoff = quantile(Laid_Off_Count, 0.25, na.rm = TRUE),
           High_Cutoff = quantile(Laid_Off_Count, 0.75, na.rm = TRUE),
           Layoff_Category = case_when(
               Laid_Off_Count <= Low_Cutoff ~ "Low Layoffs",
               Laid_Off_Count > Low_Cutoff & Laid_Off_Count <= High_Cutoff ~ "Medium Layoffs",
               Laid_Off_Count > High_Cutoff ~ "High Layoffs"
           ))

stage_mapping <- list(
    "Seed" = "Early Stage",
    "Series A" = "Early Stage",
    "Series B" = "Mid Stage",
    "Series C" = "Mid Stage",
    "Series D" = "Mid Stage",
    "Series E" = "Late Stage",
    "Series F" = "Late Stage",
    "Series G" = "Late Stage",
    "Series H" = "Late Stage",
    "Series I" = "Late Stage",
    "Post-IPO" = "Established",
    "Acquired" = "Established",
    "Private Equity" = "Established",
    "Subsidiary" = "Established",
    "Unknown" = "Unknown"
)

# Create a new column with the recategorized stages
layoffs_by_month$Stage_Categorized <- unlist(stage_mapping[layoffs_by_month$Stage])

# List of U.S. cities (excluding international cities)
us_cities <- c("Denver", "SF Bay Area", "New York City", "Austin", "Los Angeles",
               "Lehi", "Seattle", "Pittsburgh", "Phoenix", "Detroit", "Portland",
               "Boston", "San Diego", "Boulder", "Miami", "Atlanta", "Corvallis",
               "Norwalk", "Milwaukee", "Kansas City", "Salt Lake City", "Raleigh",
               "Chicago", "Santa Barbara", "Tampa Bay", "Washington D.C.",
               "Baltimore", "Dallas", "Cincinnati", "Philadelphia", "Nashua",
               "Madison", "Minneapolis", "Boise", "Columbus", "Oxford",
               "Wilmington", "Nashville", "Reno", "Burlington", "Sacramento",
               "Dover", "Logan", "Nebraska City", "San Luis Obispo",
               "Indianapolis", "Stamford", "Spokane", "Bend", "Las Vegas",
               "Santa Fe", "Missoula", "Ann Arbor")

# Filter the dataset to keep only U.S. cities
layoffs_by_month <- layoffs_by_month[layoffs_by_month$Location_HQ %in% us_cities, ]

# List unique values
unique(layoffs_by_month$Region)


west_cities <- c("Denver", "SF Bay Area", "Los Angeles", "Lehi", "Seattle", "Phoenix",
                 "Portland", "San Diego", "Boulder", "Salt Lake City", "Santa Barbara",
                 "Reno", "Sacramento", "San Luis Obispo", "Spokane", "Bend", "Las Vegas",
```

```
                  "Logan")
northeast_cities <- c("New York City", "Pittsburgh", "Boston", "Norwalk", "Philadelphia",
                      "Nashua", "Burlington")
south_cities <- c("Austin", "Miami", "Atlanta", "Corvallis", "Raleigh", "Tampa Bay",
                  "Washington D.C.", "Baltimore", "Dallas", "Nashville", "Santa Fe",
                  "Stamford")
midwest_cities <- c("Detroit", "Milwaukee", "Kansas City", "Chicago", "Cincinnati",
                    "Madison", "Minneapolis", "Boise", "Columbus", "Oxford",
                    "Wilmington", "Indianapolis", "Nebraska City", "Missoula",
                    "Ann Arbor", "Dover")


# Update city_to_region mapping
city_to_region <- list(
  "West" = west_cities,
  "Northeast" = northeast_cities,
  "South" = south_cities,
  "Midwest" = midwest_cities
)

# Function to map cities to regions
get_region <- function(city) {
  for (region in names(city_to_region)) {
    if (city %in% city_to_region[[region]]) {
      return(region)
    }
  }
  return(NA)
}
# Reapply the mapping
layoffs_by_month$Region <- sapply(layoffs_by_month$Location_HQ, get_region)

# Check the distribution of regions and NA values
table(layoffs_by_month$Region)

# Generate a grid of unique combinations of cities and stages
unique_cities <- unique(layoffs_by_month$Location_HQ)
grid <- expand.grid(
  Stage_Categorized = unique(layoffs_by_month$Stage_Categorized),
  Location_HQ = unique_cities
)
# Generate a grid of unique combinations of stages and regions
grid <- expand.grid(
  Stage_Categorized = unique(layoffs_by_month$Stage_Categorized),
  Region = unique(layoffs_by_month$Region)
)
mlr_model <- multinom(Layoff_Category ~ Stage_Categorized + Region, data = layoffs_by_month)
predictions <- predict(mlr_model, newdata = layoffs_by_month)

# Ensure actual labels are factors
layoffs_by_month$Layoff_Category <- factor(layoffs_by_month$Layoff_Category)

# Ensure predicted labels are factors
```

```r
predictions <- factor(predictions, levels = levels(layoffs_by_month$Layoff_Category))



# Predict probabilities using the region data
predictions <- predict(mlr_model, newdata = grid, type = "probs")

# Combine the predictions back into the grid
grid <- cbind(grid, predictions)

# Convert to long format for plotting
library(tidyr)
long_grid <- gather(grid, Layoff_Category, Probability, -Stage_Categorized, -Region)

# Plot interaction effects using geom_line and the Region column
library(ggplot2)
ggplot(long_grid, aes(x = Stage_Categorized, y = Probability, color = Region, group = Region)) +
  geom_line(size = 0.9, position = position_dodge(width = 0.2), alpha = 0.7) +
  facet_wrap(~ Layoff_Category) +
  theme_minimal() +
  labs(title = "Interaction Effects on Predicted Probability of Layoffs by Region", x = "Stage Categori
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
# Layoff count by year
layoffs_by_year <- layoffs_by_month %>%
  group_by(Year) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count))

plot_layoffs_by_year <- ggplot(layoffs_by_year, aes(x = as.factor(Year), y = Total_Layoffs)) +
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Total Layoff Count by Year", x = "Year", y = "Total Layoffs") +
  theme_minimal()
plot_layoffs_by_year

# Layoff count by month
layoffs_by_months <- layoffs_by_month %>%
  group_by(Month) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count))

plot_layoffs_by_month <- ggplot(layoffs_by_months, aes(x = as.factor(Month), y = Total_Layoffs)) +
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Total Layoff Count by Month", x = "Month", y = "Total Layoffs") +
  theme_minimal() +
  scale_x_discrete(labels = c("01" = "Jan", "02" = "Feb", "03" = "Mar", "04" = "Apr",
                              "05" = "May", "06" = "Jun", "07" = "Jul", "08" = "Aug",
                              "09" = "Sep", "10" = "Oct", "11" = "Nov", "12" = "Dec"))
plot_layoffs_by_month

# Layoff count by state
layoffs_by_state <- layoffs_by_month %>%
  group_by(Region) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count))

plot_layoffs_by_state <- ggplot(layoffs_by_state, aes(x = reorder(Region, -Total_Layoffs), y = Total_La
```

```
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Total Layoff Count by Region", x = "Region", y = "Total Layoffs") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme_minimal()
plot_layoffs_by_state

# Layoff count by company
layoffs_by_company <- layoffs_by_month %>%
  group_by(Company) %>%
  summarize(Total_Layoffs = sum(Laid_Off_Count)) %>%
  arrange(desc(Total_Layoffs))  # Sort in descending order of total layoffs

# Select top companies
top_companies <- head(layoffs_by_company, 10)  # Top 10 companies

plot_top_layoffs_by_company <- ggplot(top_companies, aes(x = reorder(Company, Total_Layoffs), y = Total_
  geom_bar(stat = "identity", fill = "dark green") +
  labs(title = "Top Companies by Total Layoff Count", x = "Company", y = "Total Layoffs") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  theme_minimal()
plot_top_layoffs_by_company
layoffs_by_month$Month <- as.numeric(layoffs_by_month$Month)

layoffs_by_month$Layoff_Category <- factor(layoffs_by_month$Layoff_Category)
numeric_features <- layoffs_by_month[, c("Laid_Off_Count", "Month")]
target <- layoffs_by_month$Layoff_Category
scaled_data <- scale(numeric_features)

# Create a split index
set.seed(42)
index <- sample(1:nrow(layoffs_by_month), size = 0.7 * nrow(layoffs_by_month))

# Create training and testing sets
train_data <- scaled_data[index, ]
train_target <- target[index]
test_data <- scaled_data[-index, ]
test_target <- target[-index]

# Apply KNN, choose an appropriate value for k
k <- 5  # Adjust based on your data
predicted_target <- knn(train = train_data, test = test_data, cl = train_target, k = k)
# Measure the accuracy of the predictions
accuracy <- sum(predicted_target == test_target) / length(test_target)
cat("Accuracy:", accuracy)

summary(accuracy)

# Create a data frame for plotting
plot_data <- data.frame(Actual = test_target, Predicted = predicted_target)

# Scatter plot to show the results
ggplot(plot_data, aes(x = Actual, y = Predicted, color = Predicted)) +
  geom_jitter(width = 0.1, height = 0.1) +
```

```r
  labs(title = "KNN Predictions vs Actual", x = "Actual Category", y = "Predicted Category", color = "Pr
  theme_minimal() +
  theme(legend.position = "bottom") +
  scale_color_manual(values = c("red", "green", "blue"))  # Adjust colors for each category


# Example accuracies, replace with actual values
knn_accuracy <- 0.8788927  # Example accuracy for KNN
mlr_accuracy <- 0.5218   # Example accuracy for Multinomial Logistic Regression

# Create a data frame
results_df <- data.frame(
  Model = c("KNN", "MLR"),
  Accuracy = c(knn_accuracy, mlr_accuracy)
)
# Generate table
accuracy_table <- kable(results_df, format = "html", col.names = c("Model", "Accuracy"), align = c('l',

# Print table
print(accuracy_table)
```