Introduction to linear regression

The Human Freedom Index is a report that attempts to summarize the idea of "freedom" through a bunch of different variables for many countries around the globe. It serves as a rough objective measure for the relationships between the different types of freedom - whether it's political, religious, economical or personal freedom - and other social and economic circumstances. The Human Freedom Index is an annually co-published report by the Cato Institute, the Fraser Institute, and the Liberales Institut at the Friedrich Naumann Foundation for Freedom.

In this lab, you'll be analyzing data from Human Freedom Index reports from 2008-2016. Your aim will be to summarize a few of the relationships within the data both graphically and numerically in order to find which variables can help tell a story about freedom.

Getting Started

Load packages

In this lab, you will explore and visualize the data using the **tidyverse** suite of packages. The data can be found in the companion package for OpenIntro resources, **openintro**.

Let's load the packages.

```
library(tidyverse)
library(openintro)
data('hfi', package='openintro')
```

The data

The data we're working with is in the openintro package and it's called `hfi`, short for Human Freedom Index.

1. What are the dimensions of the dataset?

```
dim(hfi)
```

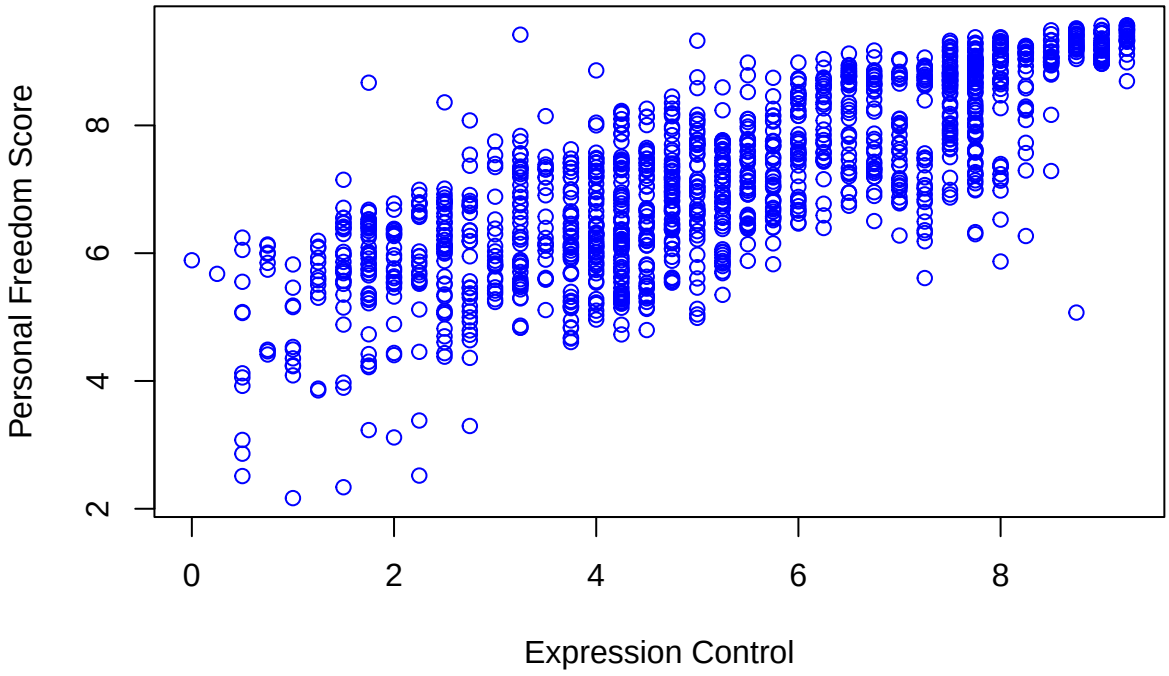
```
## [1] 1458  123
```

The dimensions of the dataset hfi are 1458 rows and 123 columns.

2. What type of plot would you use to display the relationship between the personal freedom score, `pf_score`, and one of the other numerical variables? Plot this relationship using the variable `pf_expression_control` as the predictor. Does the relationship look linear? If you knew a country's `pf_expression_control`, or its score out of 10, with 0 being the most, of political pressures and controls on media content, would you be comfortable using a linear model to predict the personal freedom score?

I would utilize a scatter plot to illustrate the relationship between the personal freedom score, pf_score, and another numerical variable.

```
plot(hfi$pf_expression_control, hfi$pf_score,
     xlab = "Expression Control", ylab = "Personal Freedom Score", col = "blue")
```



The relationship seems to be mostly linear. If we had a country's pf_expression_control score, which is out of 10 (with 0 representing the most political pressure and media control), I would feel somewhat confident in using a linear model to predict the personal freedom score.

If the relationship looks linear, we can quantify the strength of the relationship with the correlation coefficient.

```
hfi %>%
  summarise(cor(pf_expression_control, pf_score, use = "complete.obs"))
```

```
## # A tibble: 1 x 1
##   `cor(pf_expression_control, pf_score, use = "complete.obs")`
##                                                          <dbl>
## 1                                                        0.796
```

Here, we set the `use` argument to "complete.obs" since there are some observations of NA.

Sum of squared residuals

In this section, you will use an interactive function to investigate what we mean by "sum of squared residuals". You will need to run this function in your console, not in your markdown document. Running the function also requires that the `hfi` dataset is loaded in your environment.

Think back to the way that we described the distribution of a single variable. Recall that we discussed characteristics such as center, spread, and shape. It's also useful to be able to describe the relationship of two numerical variables, such as `pf_expression_control` and `pf_score` above.

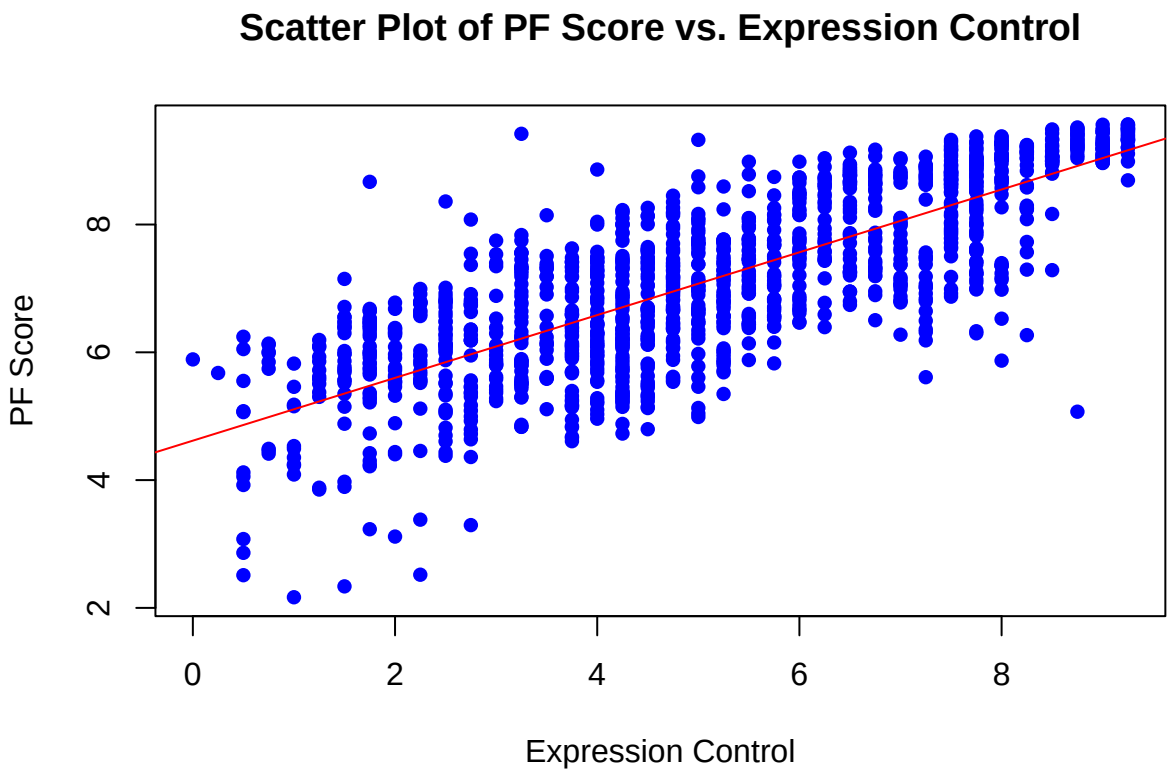
3. Looking at your plot from the previous exercise, describe the relationship between these two variables. Make sure to discuss the form, direction, and strength of the relationship as well as any unusual observations.

There is a clear connection between the two variables, with a linear relationship present. The correlation is moderately strong and positive, meaning that as pf_expression_control rises, pf_score also increases. It's important to note, however, that there are outliers or points where the predicted values would significantly differ from the observed ones.

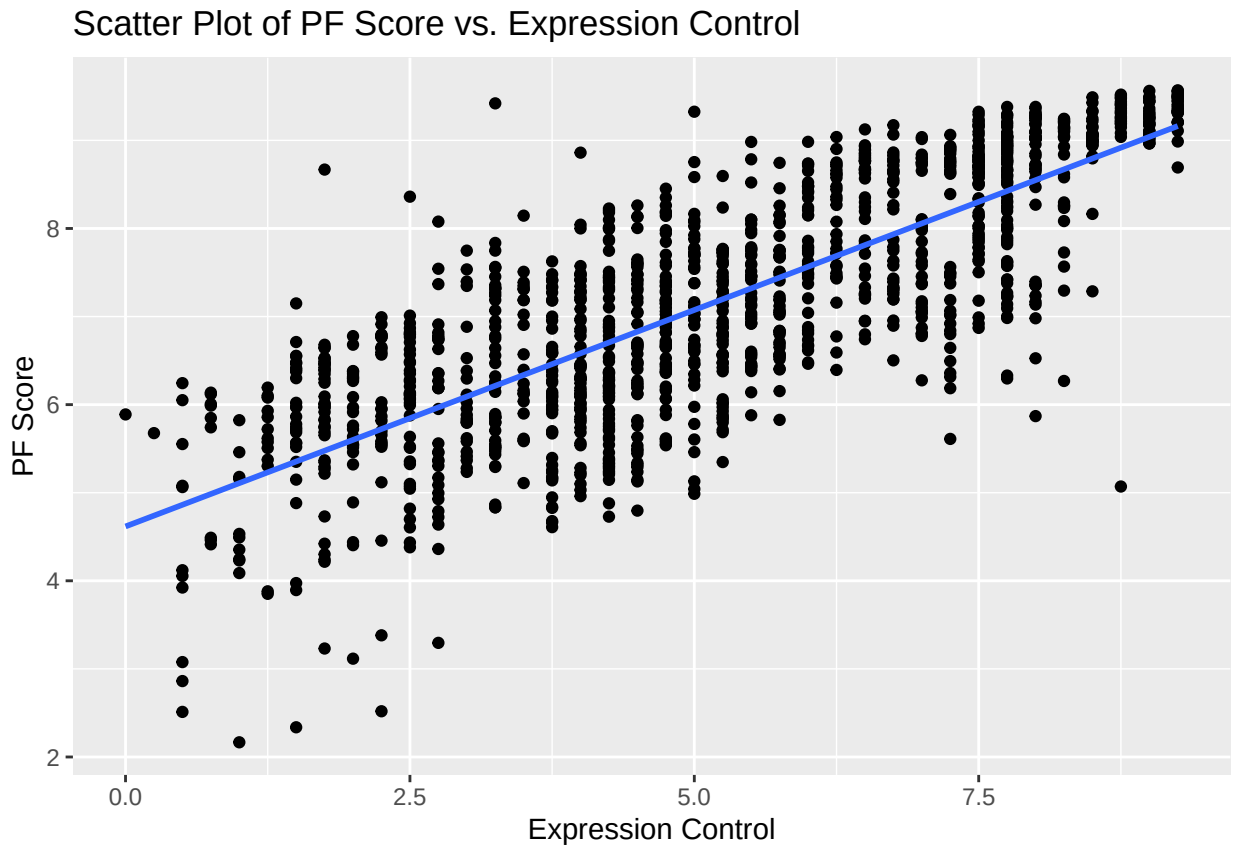
Just as you've used the mean and standard deviation to summarize a single variable, you can summarize the relationship between these two variables by finding the line that best follows their association. Use the following interactive function to select the line that you think does the best job of going through the cloud of points.

```
plot_ss <- function(x, y, xlab = "Expression Control", ylab = "PF Score", title = "Scatter Plot of PF S
  # Fit the linear model
  model <- lm(y ~ x)

  # Calculating slope, intercept, and sum of squares
  slope <- coef(model)[2]
  intercept <- coef(model)[1]
  sum_of_squares <- sum(residuals(model)^2)

  # Printing the slope, intercept, and sum of squares
  cat("Slope:", slope, "\n")
  cat("Intercept:", intercept, "\n")
  cat("Sum of Squares:", sum_of_squares, "\n")

  # Creating the scatter plot
  plot(x, y,
       xlab = xlab,
       ylab = ylab,
       main = title,
       col = "blue",
       pch = 16)  # Custom point character

  # Add the regression line
  abline(model, col = "red")
}

# Filtering the data to remove rows with NA values
hfi_clean <- hfi %>% filter(complete.cases(pf_expression_control, pf_score))

# Calling the custom plot function with filtered data
plot_ss(hfi_clean$pf_expression_control, hfi_clean$pf_score)
```

```
## Slope: 0.4914312
## Intercept: 4.617074
## Sum of Squares: 952.1532
```




```
library(ggplot2)

ggplot(data = hfi, aes(x = pf_expression_control, y = pf_score)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  labs(x = "Expression Control", y = "PF Score",
       title = "Scatter Plot of PF Score vs. Expression Control")
```




```
# This will only work interactively (i.e. will not show in the knitted document)
hfi <- hfi %>% filter(complete.cases(pf_expression_control, pf_score))
DATA606::plot_ss(x = hfi$pf_expression_control, y = hfi$pf_score)
```

After running this command, you'll be prompted to click two points on the plot to define a line. Once you've done that, the line you specified will be shown in black and the residuals in blue. Note that there are 30 residuals, one for each of the 30 observations. Recall that the residuals are the difference between the observed values and the values predicted by the line:

$$e_i = y_i - \hat{y}_i$$

The most common way to do linear regression is to select the line that minimizes the sum of squared residuals. To visualize the squared residuals, you can rerun the plot command and add the argument `showSquares = TRUE`.

```
# Creating a subset of the hfi data frame with specified columns
hfi1 <- hfi[c("pf_score", "pf_expression_control")] %>%
  drop_na()  # Drop rows with NAs
```

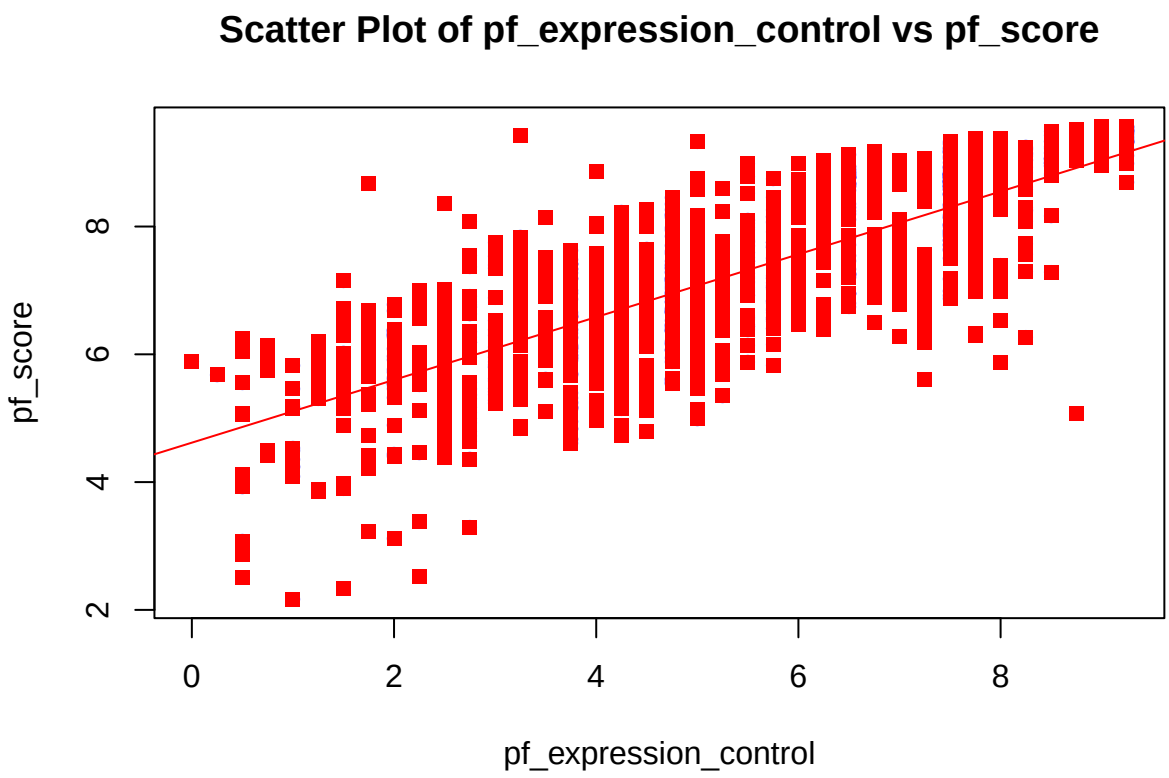
```
# Now we can use hfi1 in our analysis
lm_model <- lm(pf_score ~ pf_expression_control, data = hfi1)
summary(lm_model)
```

```
##
```

```
## Call:
## lm(formula = pf_score ~ pf_expression_control, data = hfi1)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8467 -0.5704  0.1452  0.6066  3.2060
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.61707    0.05745   80.36   <2e-16 ***
## pf_expression_control  0.49143    0.01006   48.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8318 on 1376 degrees of freedom
## Multiple R-squared:  0.6342, Adjusted R-squared:  0.634
## F-statistic:  2386 on 1 and 1376 DF,  p-value: < 2.2e-16
```

```r
plot_ss <- function(x, y, data, showSquares = FALSE) {
  # Fit the linear model
  model <- lm(data[[y]] ~ data[[x]])

  # Calculating slope, intercept, and sum of squares
  slope <- coef(model)[2]
  intercept <- coef(model)[1]
  sum_of_squares <- sum(residuals(model)^2)

  # Printing the slope, intercept, and sum of squares
  cat("Slope:", slope, "\n")
  cat("Intercept:", intercept, "\n")
  cat("Sum of Squares:", sum_of_squares, "\n")

  # Scatter plot
  plot(data[[x]], data[[y]],
       xlab = x, ylab = y,
       main = paste("Scatter Plot of", x, "vs", y),
       col = "blue", pch = 16)

  # Adding the regression line
  abline(model, col = "red")

  # Optionally add squares if showSquares is TRUE
  if (showSquares) {
    points(data[[x]], data[[y]], pch = 15, col = "red")
  }
}

# Using the modified function
plot_ss(x = "pf_expression_control", y = "pf_score", data = hfi1, showSquares = TRUE)
```

```
## Slope: 0.4914312
## Intercept: 4.617074
## Sum of Squares: 952.1532
```

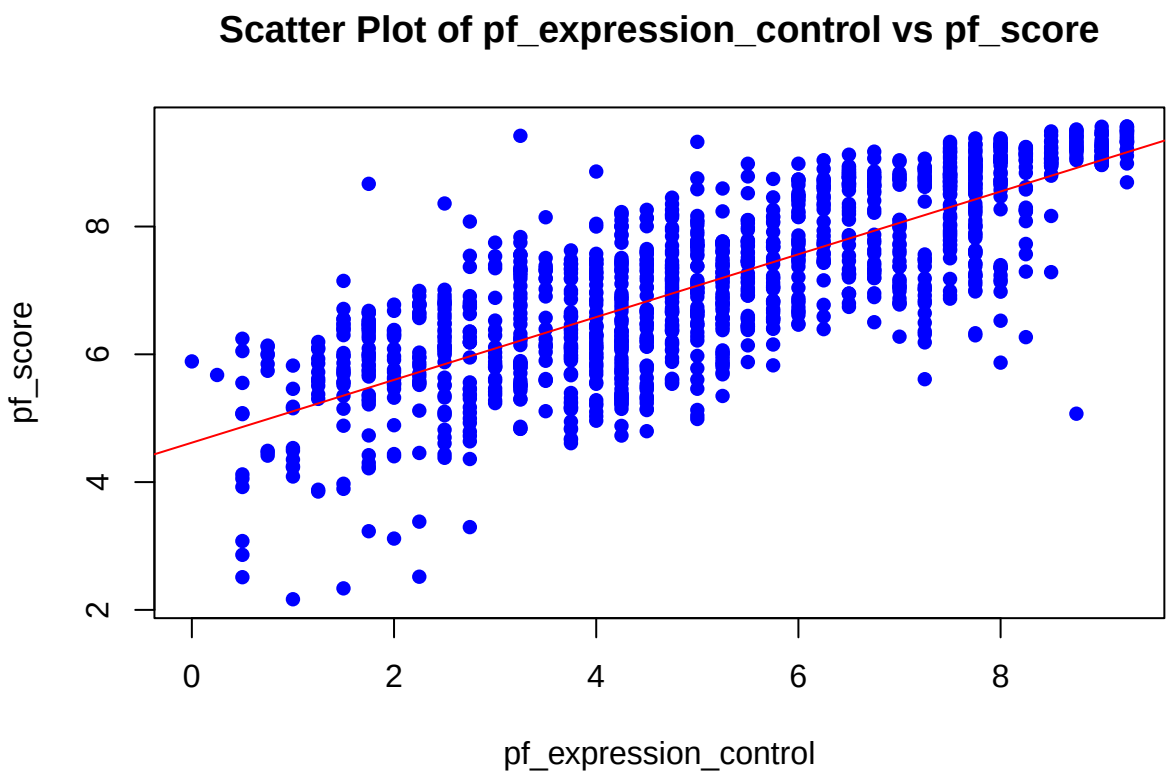


Note that the output from the `plot_ss` function provides you with the slope and intercept of your line as well as the sum of squares.

4. Using `plot_ss`, choose a line that does a good job of minimizing the sum of squares. Run the function several times. What was the smallest sum of squares that you got? How does it compare to your neighbors?

```
plot_ss(x = "pf_expression_control", y = "pf_score", data = hfi1)
```

```
## Slope: 0.4914312
## Intercept: 4.617074
## Sum of Squares: 952.1532
```



The smallest sum of squares is 952.153. The model with the lowest sum of squares generally provides a better fit to the data.

The linear model

It is rather cumbersome to try to get the correct least squares line, i.e. the line that minimizes the sum of squared residuals, through trial and error. Instead, you can use the `lm` function in R to fit the linear model (a.k.a. regression line).

```
m1 <- lm(pf_score ~ pf_expression_control, data = hfi)
```

The first argument in the function `lm` is a formula that takes the form `y ~ x`. Here it can be read that we want to make a linear model of `pf_score` as a function of `pf_expression_control`. The second argument specifies that R should look in the `hfi` data frame to find the two variables.

The output of `lm` is an object that contains all of the information we need about the linear model that was just fit. We can access this information using the summary function.

```
summary(m1)
```

```
##
## Call:
## lm(formula = pf_score ~ pf_expression_control, data = hfi)
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -3.8467 -0.5704  0.1452  0.6066  3.2060
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)            4.61707    0.05745   80.36   <2e-16 ***
## pf_expression_control  0.49143    0.01006   48.85   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8318 on 1376 degrees of freedom
##   (80 observations deleted due to missingness)
## Multiple R-squared:  0.6342, Adjusted R-squared:  0.634
## F-statistic:  2386 on 1 and 1376 DF,  p-value: < 2.2e-16
```

Let's consider this output piece by piece. First, the formula used to describe the model is shown at the top. After the formula you find the five-number summary of the residuals. The "Coefficients" table shown next is key; its first column displays the linear model's y-intercept and the coefficient of `pf_expression_control`. With this table, we can write down the least squares regression line for the linear model:

$$\hat{y} = 4.61707 + 0.49143 \times pf\_expression\_control$$

One last piece of information we will discuss from the summary output is the Multiple R-squared, or more simply, R^2. The R^2 value represents the proportion of variability in the response variable that is explained by the explanatory variable. For this model, 63.42% of the variability in runs is explained by at-bats.

5. Fit a new model that uses `pf_expression_control` to predict `hf_score`, or the total human freedom score. Using the estimates from the R output, write the equation of the regression line. What does the slope tell us in the context of the relationship between human freedom and the amount of political pressure on media content?

```r
# Fitting a linear model with hf_score as the response and pf_expression_control as the predictor.
linear_model <- lm(formula = hf_score ~ pf_expression_control, data = hfi)

# Displaying the summary of the linear model
summary(linear_model)
```

```
##
## Call:
## lm(formula = hf_score ~ pf_expression_control, data = hfi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6198 -0.4908  0.1031  0.4703  2.2933
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.153687   0.046070  111.87   <2e-16 ***
## pf_expression_control 0.349862   0.008067   43.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.667 on 1376 degrees of freedom
##   (80 observations deleted due to missingness)
## Multiple R-squared:  0.5775, Adjusted R-squared:  0.5772
## F-statistic:  1881 on 1 and 1376 DF,  p-value: < 2.2e-16
```

Equation: The relationship can be expressed as: hf_score = 5.1537 + 0.3499 × pf_expression_control hf_score = 5.1537 + 0.3499 × pf_expression_control

Slope: This indicates that for every additional unit increase in the political pressure on media content score, we anticipate an increase of 0.3499 in the human freedom score.

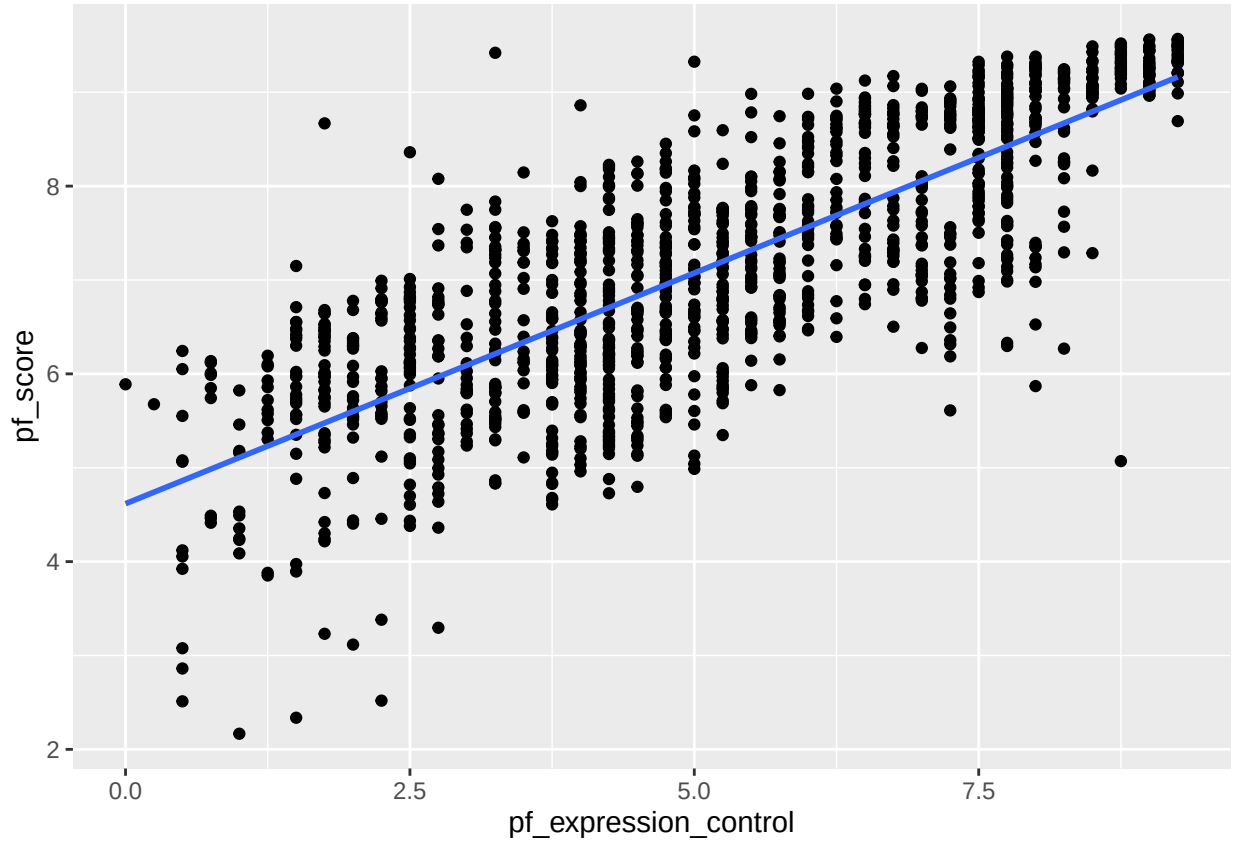
Intercept: When there is no political pressure on media, the human freedom score is predicted to be 5.1537.

Insert your answer here

Prediction and prediction errors

Let's create a scatterplot with the least squares line for `m1` laid on top.

```
ggplot(data = hfi, aes(x = pf_expression_control, y = pf_score)) +
  geom_point() +
  stat_smooth(method = "lm", se = FALSE)
```



Here, we are literally adding a layer on top of our plot. `geom_smooth` creates the line by fitting a linear model. It can also show us the standard error `se` associated with our line, but we'll suppress that for now.

This line can be used to predict y at any value of x. When predictions are made for values of x that are beyond the range of the observed data, it is referred to as *extrapolation* and is not usually recommended.

However, predictions made within the range of the data are more reliable. They're also used to compute the residuals.

6. If someone saw the least squares regression line and not the actual data, how would they predict a country's personal freedom school for one with a 6.7 rating for `pf_expression_control`? Is this an overestimate or an underestimate, and by how much? In other words, what is the residual for this prediction?

Calculating pf_score6 using the linear equation

```
# Defining the political expression control value
pf_exp_control <- 6.7

# Calculating pf_score6 using the linear equation
pf_score6 <- 4.61707 + (0.49143 * pf_exp_control)

# Displaying the result
pf_score6
```

```
## [1] 7.909651
```

Retrieving observed values of pf_score

```
# Retrieving observed values of pf_score for a pf_expression_control rating of 6.7
hfi %>%
  filter(pf_expression_control == 6.7) %>%
  select(pf_score)
```

```
## # A tibble: 0 x 1
## # i 1 variable: pf_score <dbl>
```

No observed pf_score corresponds to a pf_expression_control of 6.7, so I'll use nearest pf_score of 7.43 for a rating of 6.75.

```
# Calculating the residual
residual <- 7.43 - 7.91

# Displaying the residual value
residual
```

```
## [1] -0.48
```

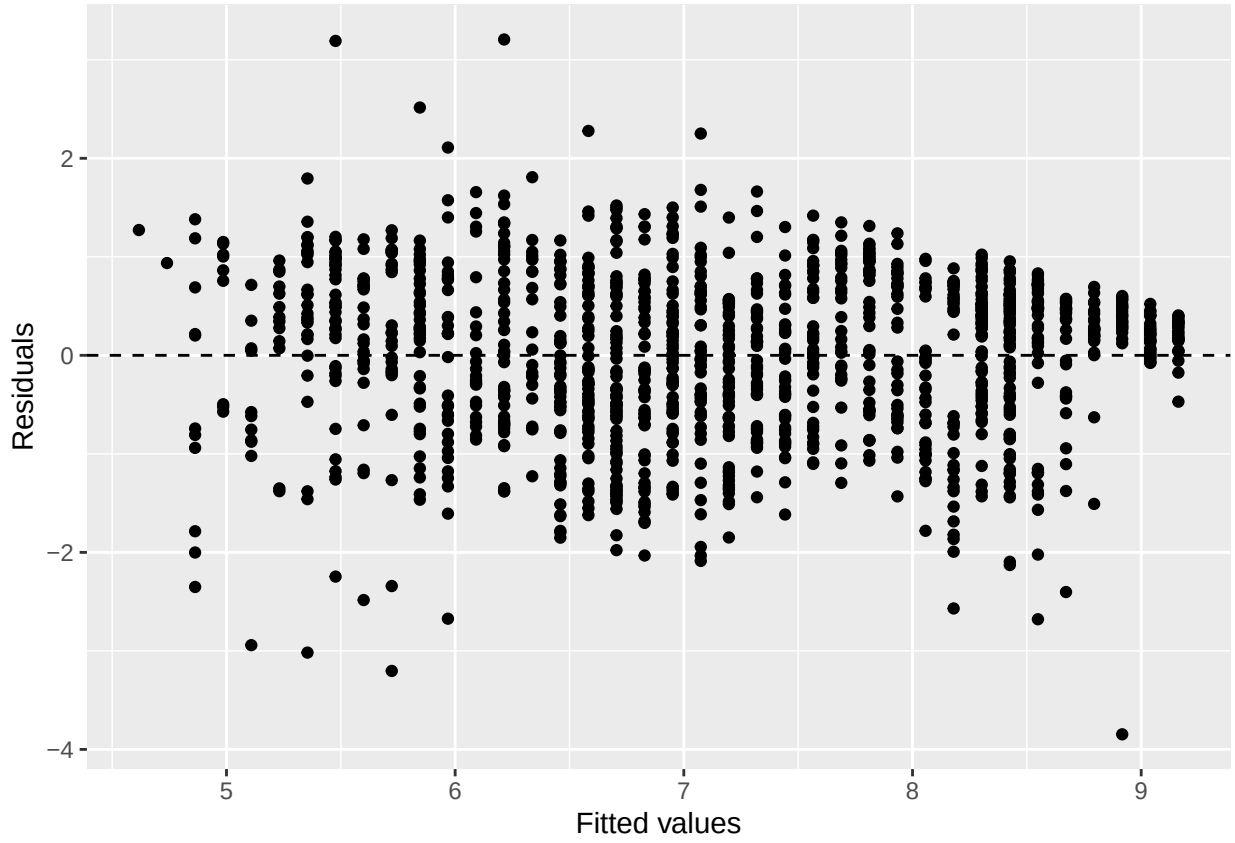
The prediction was higher than the actual value by -0.48.

Model diagnostics

To assess whether the linear model is reliable, we need to check for (1) linearity, (2) nearly normal residuals, and (3) constant variability.

Linearity: You already checked if the relationship between `pf_score` and 'pf_expression_control' is linear using a scatterplot. We should also verify this condition with a plot of the residuals vs. fitted (predicted) values.

```
ggplot(data = m1, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") +
  xlab("Fitted values") +
  ylab("Residuals")
```



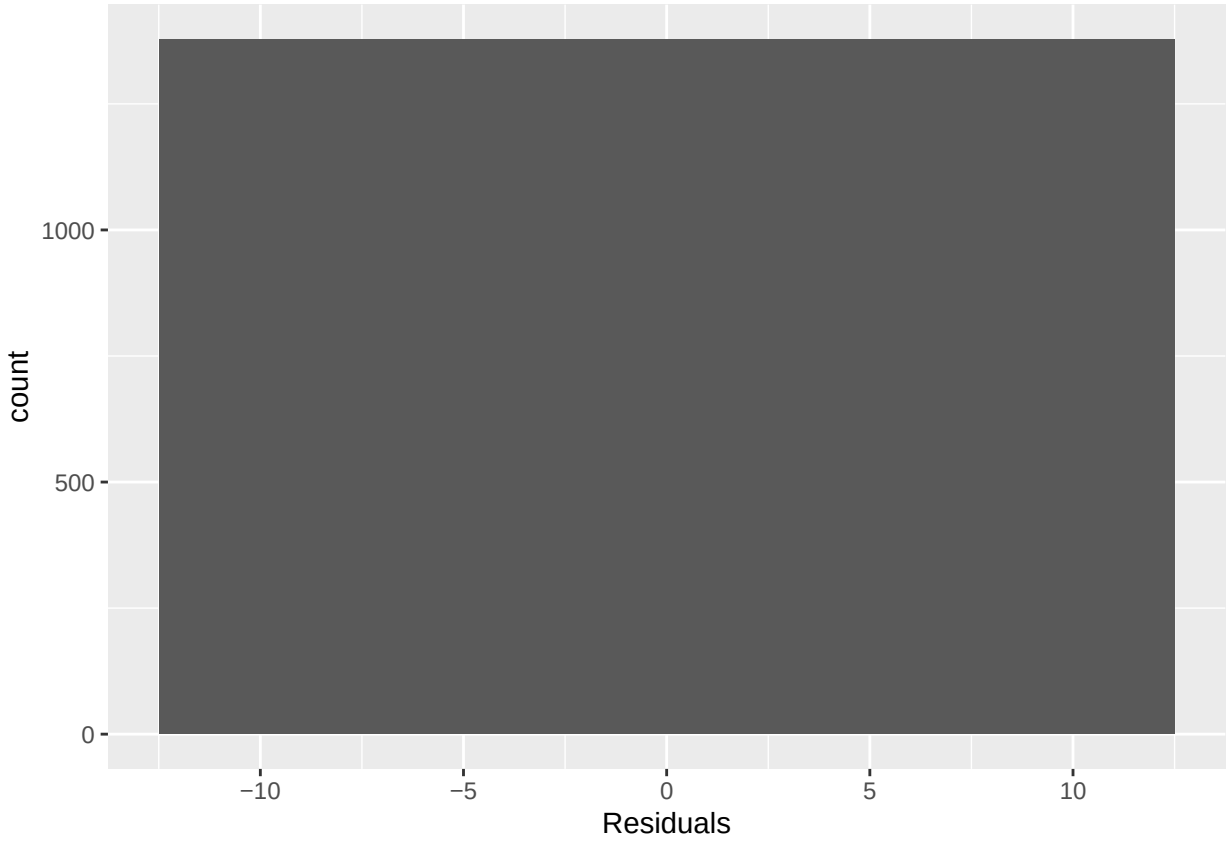
Notice here that `m1` can also serve as a data set because stored within it are the fitted values ($\hat{y}$) and the residuals. Also note that we're getting fancy with the code here. After creating the scatterplot on the first layer (first line of code), we overlay a horizontal dashed line at $y = 0$ (to help us check whether residuals are distributed around 0), and we also reanme the axis labels to be more informative.

7. Is there any apparent pattern in the residuals plot? What does this indicate about the linearity of the relationship between the two variables?

The residuals plot does not reveal any noticeable pattern, suggesting that a linear relationship exists between the two variables.

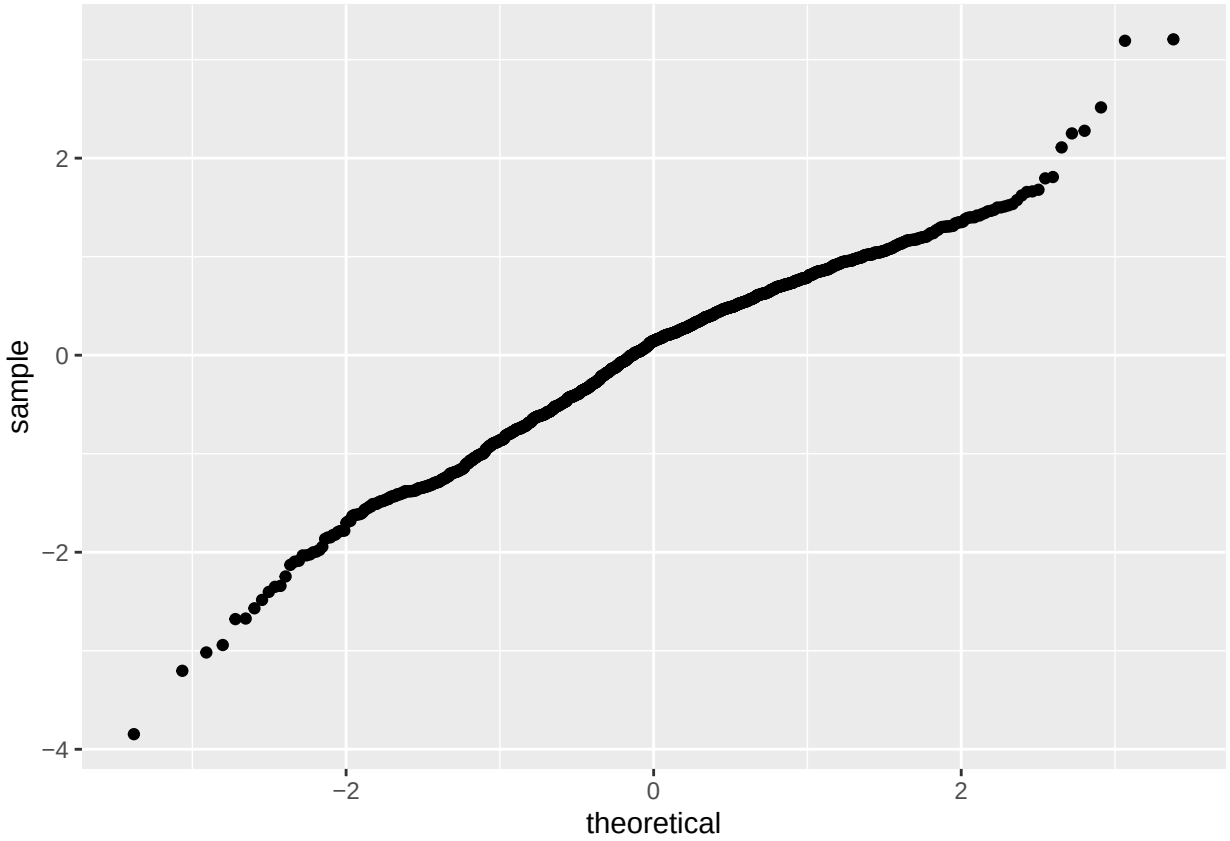
Nearly normal residuals: To check this condition, we can look at a histogram

```
ggplot(data = m1, aes(x = .resid)) +
  geom_histogram(binwidth = 25) +
  xlab("Residuals")
```



or a normal probability plot of the residuals.

```
ggplot(data = m1, aes(sample = .resid)) +
  stat_qq()
```



Note that the syntax for making a normal probability plot is a bit different than what you're used to seeing: we set `sample` equal to the residuals instead of `x`, and we set a statistical method `qq`, which stands for "quantile-quantile", another name commonly used for normal probability plots.

8. Based on the histogram and the normal probability plot, does the nearly normal residuals condition appear to be met?

Both the histogram and the normal probability plot indicate that the data distribution is approximately normal. Therefore, the condition for nearly normal residuals seems to be satisfied.

Constant variability:

9. Based on the residuals vs. fitted plot, does the constant variability condition appear to be met?

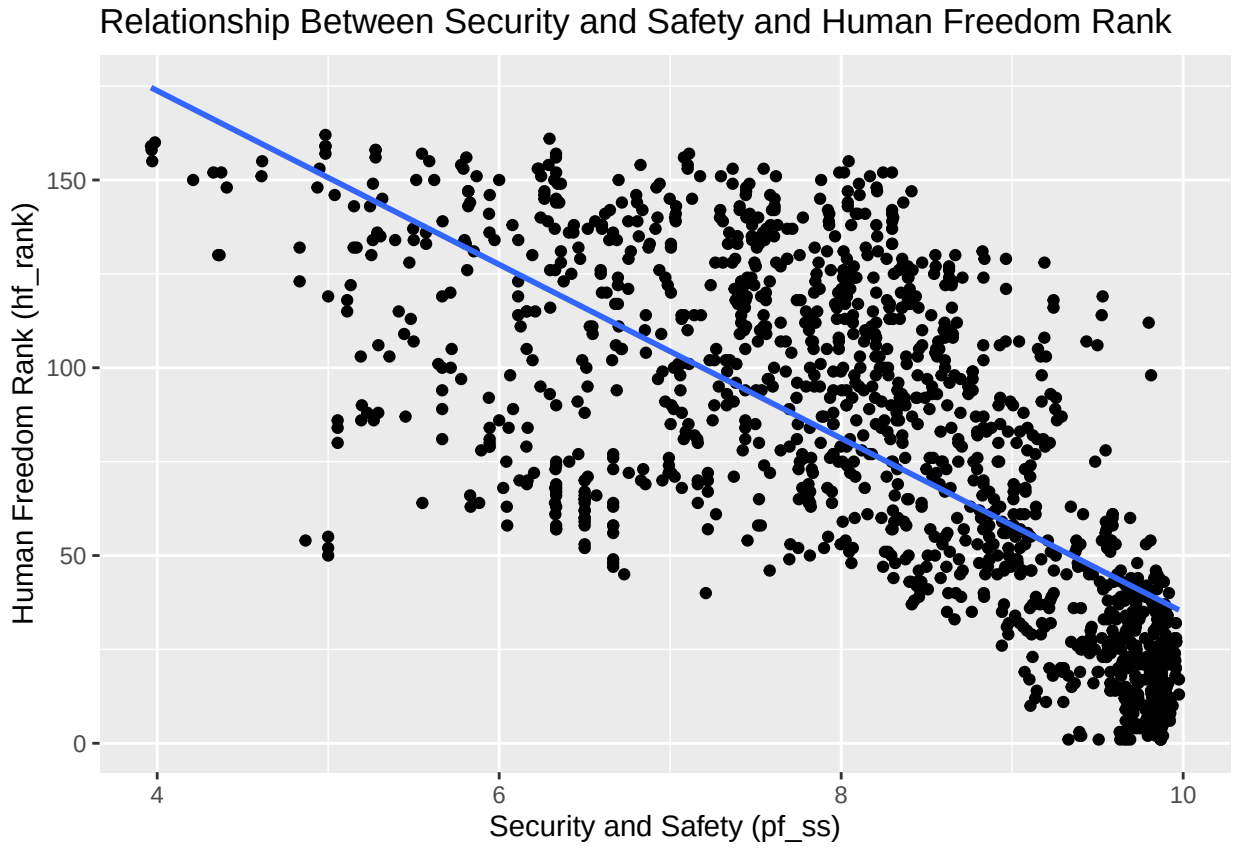
The residuals versus fitted plot reveals that the points are dispersed around zero, indicating consistent variability. Therefore, the condition for constant variability appears to be fulfilled.

More Practice

- Choose another freedom variable and a variable you think would strongly correlate with it.. Produce a scatterplot of the two variables and fit a linear model. At a glance, does there seem to be a linear relationship?

Upon initial observation, the connection between these two variables seems to be linear. There exists a negative relationship: as pf_ss rises, hf_rank falls.

```
# Predicting human freedom rank (hf_rank) using Security and Safety (pf_ss) as a predictor
library(ggplot2)

# Creating a scatter plot with a linear regression line
ggplot(data = hfi, aes(x = pf_ss, y = hf_rank)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +  # Adding a linear model fit without confidence interval
  labs(x = "Security and Safety (pf_ss)", y = "Human Freedom Rank (hf_rank)",
       title = "Relationship Between Security and Safety and Human Freedom Rank")
```



- How does this relationship compare to the relationship between `pf_expression_control` and `pf_score`? Use the R^2 values from the two model summaries to compare. Does your independent variable seem to predict your dependent one better? Why or why not?

```
# Creating a linear model predicting hf_score using pf_expression_control
model_expression_control <- lm(formula = hf_score ~ pf_expression_control, data = hfi)

# Output of the linear model (Summary)
model_summary <- summary(model_expression_control)
model_summary
```

```
##
```

```
## Call:
## lm(formula = hf_score ~ pf_expression_control, data = hfi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.6198 -0.4908  0.1031  0.4703  2.2933
##
## Coefficients:
##                       Estimate Std. Error t value Pr(>|t|)
## (Intercept)           5.153687   0.046070  111.87   <2e-16 ***
## pf_expression_control 0.349862   0.008067   43.37   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.667 on 1376 degrees of freedom
##   (80 observations deleted due to missingness)
## Multiple R-squared:  0.5775, Adjusted R-squared:  0.5772
## F-statistic:  1881 on 1 and 1376 DF,  p-value: < 2.2e-16
```

```
# Fitting a linear model to predict hf_rank using pf_ss as the predictor
model_pf_ss <- lm(hf_rank ~ pf_ss, data = hfi)

# Displaying the summary of the linear model
model_summary_pf_ss <- summary(model_pf_ss)
model_summary_pf_ss
```

```
##
## Call:
## lm(formula = hf_rank ~ pf_ss, data = hfi)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -100.628  -23.567   -3.302   23.189   77.679
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 266.3430     5.1084   52.14   <2e-16 ***
## pf_ss       -23.1430     0.6159  -37.58   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.28 on 1376 degrees of freedom
##   (80 observations deleted due to missingness)
## Multiple R-squared:  0.5065, Adjusted R-squared:  0.5061
## F-statistic:  1412 on 1 and 1376 DF,  p-value: < 2.2e-16
```

Based on the R-squared values of both models, we observe the following:

For the model involving pf_expression_control and pf_score, 57.75% of the variability in pf_score is accounted for by pf_expression_control. In contrast, the model for pf_ss and hf_rank explains only 50.65% of the variability in hf_rank based on pf_ss. This indicates that my independent variable does not predict the dependent variable as effectively, as the R-squared value in the pf_ss and hf_rank model is lower than that of the pf_expression_control and pf_score model, resulting in less explained variation

- What's one freedom relationship you were most surprised about and why? Display the model diagnostics for the regression model analyzing this relationship.

The relationship between the integrity of the legal system and the protection of property rights was the most surprising to me. I anticipated a strong correlation, where greater variability in property legal protection would be accounted for by the integrity of the legal system. However, this was not the case; the model shows a lower R-squared value compared to the previous models. For further insights, refer to the summary and diagnostic metrics of the regression analysis concerning this relationship.

```
# Fitting a linear model to predict ef_legal_protection using ef_legal_integrity as the predictor
model_legal_protection <- lm(ef_legal_protection ~ ef_legal_integrity, data = hfi)

# Displaying the summary of the linear model
model_summary_legal_protection <- summary(model_legal_protection)
model_summary_legal_protection
```

```
##
## Call:
## lm(formula = ef_legal_protection ~ ef_legal_integrity, data = hfi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3674 -0.8148  0.1089  0.9094  3.8036
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.09412    0.11409   18.36   <2e-16 ***
## ef_legal_integrity  0.56976    0.01719   33.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.226 on 1109 degrees of freedom
##   (347 observations deleted due to missingness)
## Multiple R-squared:  0.4978, Adjusted R-squared:  0.4973
## F-statistic:  1099 on 1 and 1109 DF,  p-value: < 2.2e-16
```

```
# Calculating the correlation between ef_legal_protection and ef_legal_integrity
correlation_result <- hfi %>%
  summarise(correlation = cor(ef_legal_protection, ef_legal_integrity, use = "complete.obs"))

# Displaying the correlation result
correlation_result
```

```
## # A tibble: 1 x 1
##   correlation
##         <dbl>
## 1       0.706
```

The two variables exhibit a moderate positive correlation.

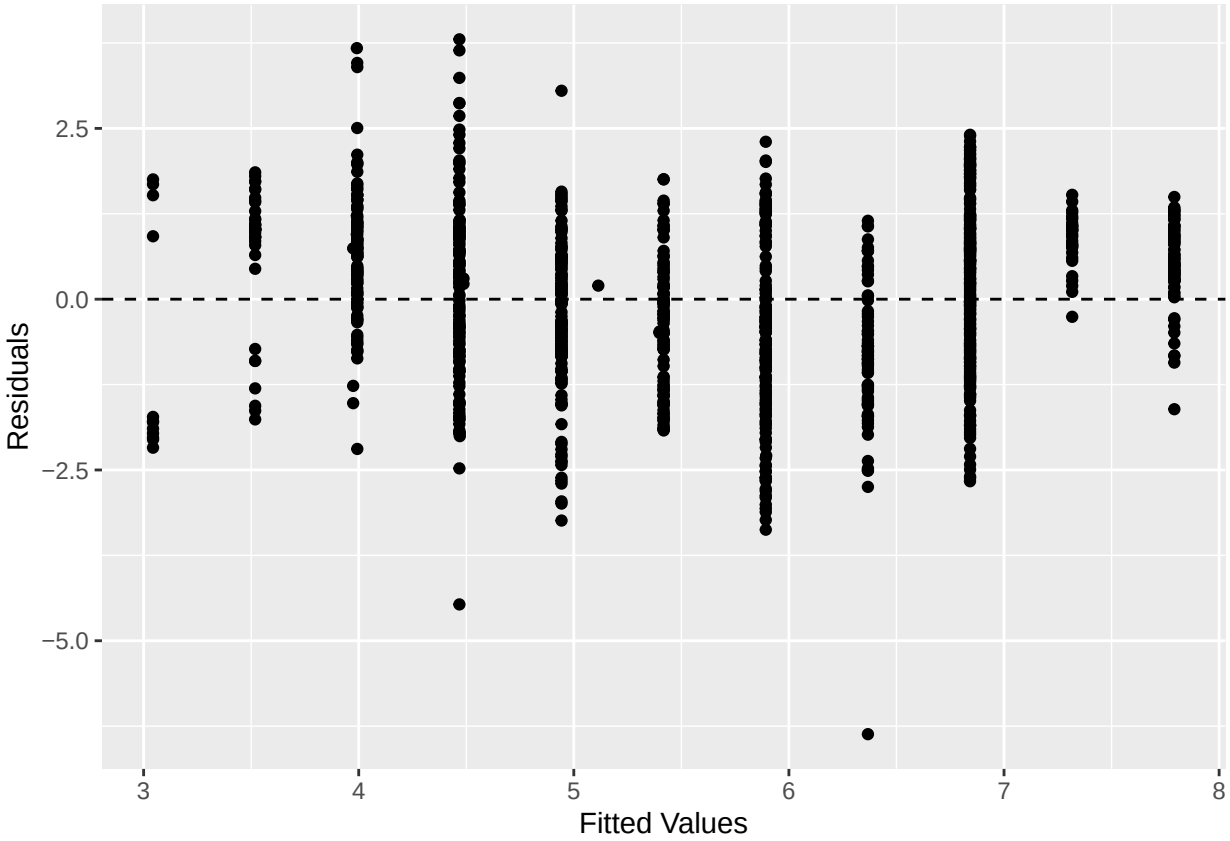
Model Diagnostics

Linearity: It's important to check this condition by examining a plot of the residuals against the fitted (predicted) values.

```r
# Creating the linear model with backticks
lmb <- lm(`ef_legal_protection` ~ `ef_legal_integrity`, data = hfi)
summary(lmb)
```

```
##
## Call:
## lm(formula = ef_legal_protection ~ ef_legal_integrity, data = hfi)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3674 -0.8148  0.1089  0.9094  3.8036
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)         2.09412    0.11409   18.36   <2e-16 ***
## ef_legal_integrity  0.56976    0.01719   33.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.226 on 1109 degrees of freedom
##   (347 observations deleted due to missingness)
## Multiple R-squared:  0.4978, Adjusted R-squared:  0.4973
## F-statistic:  1099 on 1 and 1109 DF,  p-value: < 2.2e-16
```

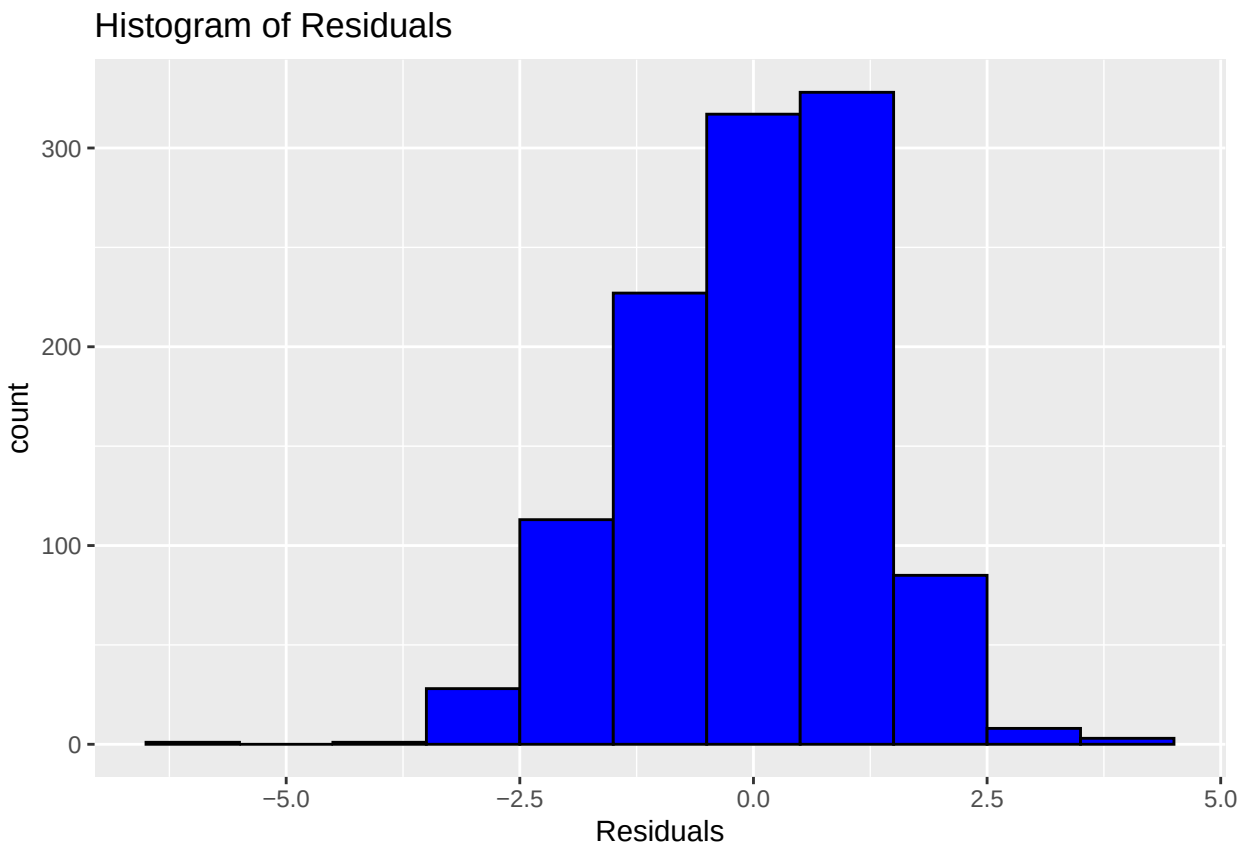
```r
# Creating a scatter plot of residuals against fitted values
ggplot(data = lmb, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_hline(yintercept = 0, linetype = "dashed") + # Adding a dashed horizontal line at y = 0
  labs(x = "Fitted Values", y = "Residuals")        # Setting labels for the axes
```



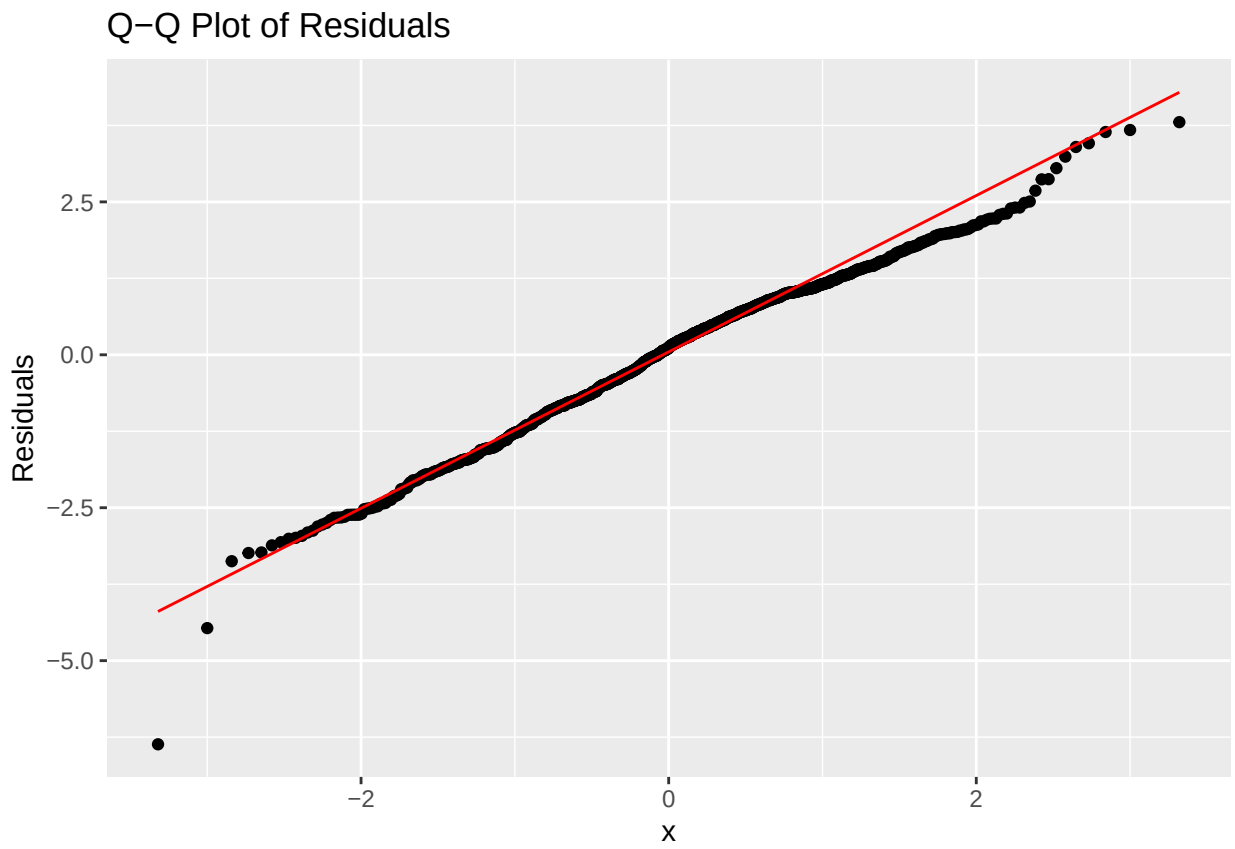
The residuals plot does not reveal any discernible pattern, suggesting that a linear relationship exists between the two variables.

To assess the condition of nearly normal residuals, we can examine a histogram or a normal probability plot of the residuals.

```
# Creating a histogram of the residuals
ggplot(data = lmb, aes(x = .resid)) +
  geom_histogram(binwidth = 1, fill = "blue", color = "black") +  # Adding fill and outline colors
  labs(x = "Residuals", title = "Histogram of Residuals")          # Setting labels for the axes and ti
```



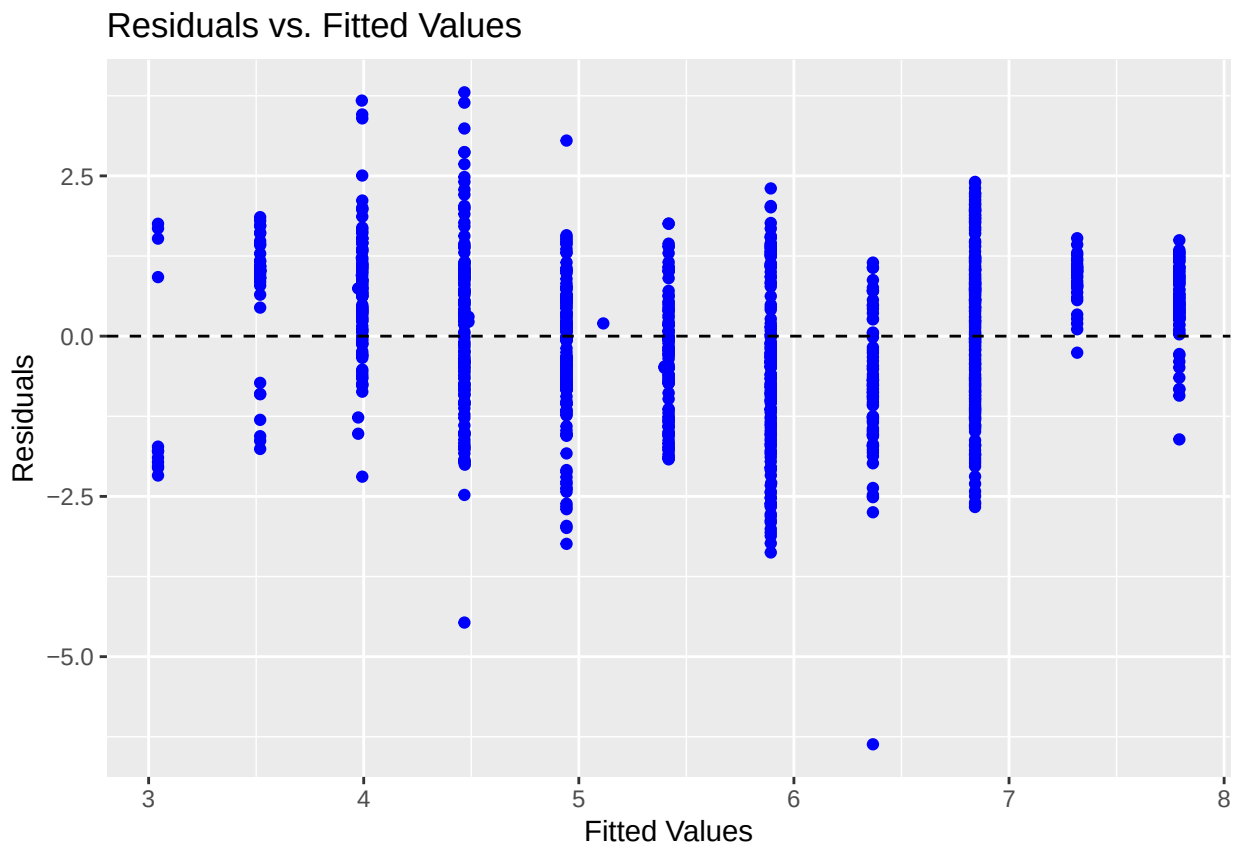

```
# Creating a Q-Q plot to assess the normality of residuals
ggplot(data = lmb, aes(sample = .resid)) +
  geom_qq() +                       # Using geom_qq() to create the Q-Q plot
  geom_qq_line(color = "red") +   # Adding a reference line to the Q-Q plot
  labs(title = "Q-Q Plot of Residuals", y = "Residuals")  # Setting the title and label the y-axis
```



Based on the histogram and the normal probability plot, we can conclude that the distribution of the data is approximately normal.

Constant variability: We will revisit the residuals vs. fitted values plot.

```
# Creating a scatter plot of residuals against fitted values
ggplot(data = lmb, aes(x = .fitted, y = .resid)) +
  geom_point(color = "blue") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = "Fitted Values", y = "Residuals", title = "Residuals vs. Fitted Values")
```



The residuals vs. fitted values plot indicates that the points are generally dispersed around zero, demonstrating a degree of constant variability.
