TMA4315: Compulsory exercise 1

Group 12: Tiago Alexandre Alcobia Pereira

13.09.2024

Contents

Part 1: Explanatory analysis of the dataset

Dataset

In this task, we have a dataset with 3987 observations and 5 variables:

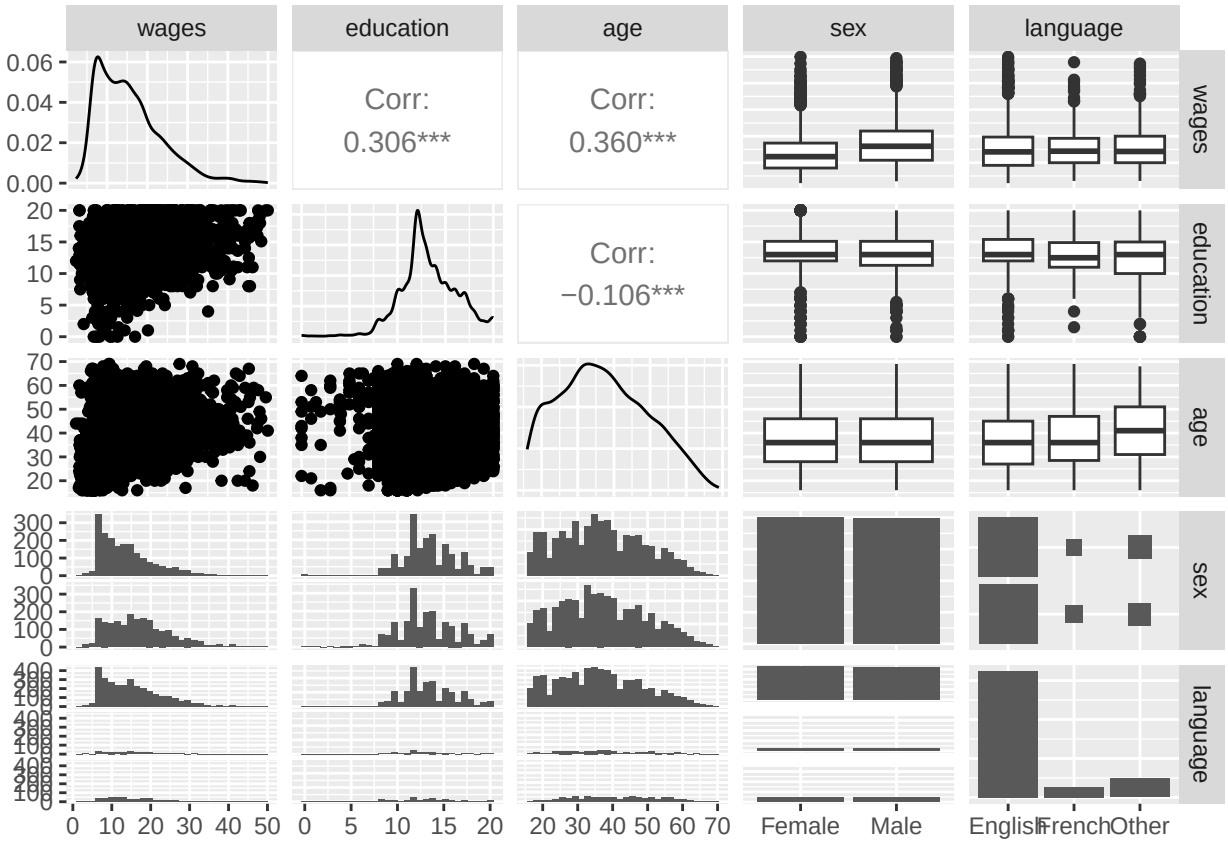
- `wages`, composite hourly wage rate from all jobs
- `education`, number of years in schooling
- `age`, in years
- `sex`, Male or Female
- `language`, English, French or Other

Then we perform what was given in the task text, which stores the data as a dataframe called SLID, and then removes all rows containing missing data:

```
library(car)
data(SLID, package = "carData")
SLID <- SLID[complete.cases(SLID),]
```

Then we draw the matrix of diagnostic plots of the variables by:

```
library(GGally)
ggpairs(data = SLID)
```



Now we evaluate the relationships, where we focus mostly on `wages` as it feels like the most relevant variable to evaluate when compared with the other variables.

For wages and languages, it's clear that English stands out as the most common language, showing a clear pattern in the histogram. Wages are similar across different languages in terms of their average minimum and maximum values. So, there's no need for more detailed analysis.

Between `sex` and `wages`, it shows that women are more likely to be found in lower-paying jobs, while men are more commonly represented in higher-paying roles. In addition, it shows that men have a bigger average income even though the maximum and minimum wages are quite similar for women and men.

The relationship between `education` and `wages` is quite similar to the relationship between `age`and `wages`, so we can evaluate just one of these to get a good idea about the other.

The correlation between age and wages is 0.360, which means that as age goes up, wages tend to go up as well. This suggests that older workers usually earn more.

From the scatter plot, we see that age varies more with lower wages, while higher wages are associated with more stable age values. This means that to earn a high wage, you generally need to be older, but being older doesn't necessarily mean you will have a high wage.

Multiple Linear Regression Analysis

In order to perform a multiple linear regression analysis, we need a classical linear model:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

Where $\mathbf{Y}$ is a column vector with n response values, $\mathbf{X}$ a $n \times p$ matrix with every observation on the rows and covariates on the columns and an intercept column with just 1s, $\mathbf{X}$ is a column vector with p model coefficients and ε is a column vector with n error values.

From this we have to make these assumptions:

- $\mathbb{E}[\varepsilon] = \mathbf{0}$
- $\text{Cov}(\varepsilon) = \mathbb{E}(\varepsilon\varepsilon^T) = \sigma^2\mathbf{I}$
- $\text{rank}(\mathbf{X}) = k + 1 = p$
- $\varepsilon \sim \mathcal{N}_n(\mathbf{0}, \sigma^2\mathbf{I})$

Where k is the number of covariates.

Part 2: Linear regression with the `mylm` package

a)

In order to fill in the missing parts in the `mylm` function, we need to estimate the coefficients, calculating them in this form:

$$\hat{\beta} = \left(\mathbf{X}^\top\mathbf{X}\right)^{-1}\mathbf{X}^\top\mathbf{Y}$$

Now we use this to create our `mylm` and compare it with `lm`:

```
library(mylm)
a_model <- mylm(wages ~ education, data = SLID)
print(a_model)
```

```
## Function Call Details:
## mylm(formula = wages ~ education, data = SLID)
##
## Model Parameters:
##   (Intercept) education
## 1    4.971691 0.7923091
```

```
a_model_test <- lm(wages ~ education, data = SLID)
print(a_model_test)
```

```
##
## Call:
## lm(formula = wages ~ education, data = SLID)
##
## Coefficients:
## (Intercept)    education
##      4.9717       0.7923
```

We can clearly see that the results are similar, which is what we wanted.

b)

We start by finding the standard error, given by:

$$\hat{\mathbf{e}} = \mathbf{Y} - X^T\hat{\beta}$$

Then, in order to find other estimates, we first need to find $\hat{\sigma}^2$, the estimate for the variance:

$$\hat{\sigma}^2 = \frac{1}{n-p}\left(\mathbf{Y} - \mathbf{X}\hat{\beta}\right)^T\left(\mathbf{Y} - \mathbf{X}\hat{\beta}\right) = \frac{\text{SSE}}{n-p}$$

Then we can find the covariance matrix:

$$\widehat{\text{Cov}}(\hat{\beta}) = \hat{\sigma}^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1} = \frac{\text{SSE}}{n-p}\left(\mathbf{X}^T\mathbf{X}\right)^{-1}$$

And we also notice that:

$$\frac{(n-p)\hat{\sigma}^2}{\sigma^2} \text{ follows a } \chi^2 \text{ distribution with } n-p \text{ degrees of freedom.}$$

From this, we find the covariance matrix of the model:

```
a_model$covariance_matrix
```

```
##             (Intercept)    education
## (Intercept)  0.28532651 -0.020338410
## education   -0.02033841  0.001524956
```

We also need a t-statistic, given by:

$$T_j = \frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}^2\sqrt{(\mathbf{X}^T\mathbf{X})^{-1}_{jj}}} \sim t_{n-p}$$

Which we approximate to:

$$T_j = \frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}^2\sqrt{(\mathbf{X}^T\mathbf{X})^{-1}_{jj}}} \sim N(0,1)$$

We can do this approximation when the number of observations n is large enough.

To test the significance of a regression coefficient, we use a z-test. This means we calculate the probability of obtaining the t-statistic or a more unusual result under the null hypothesis, which assumes that the covariate j has no effect on $\mathbf{Y}$. To do this, we:

$$z_i = \frac{\hat{\beta}_i - 0}{\sqrt{\hat{\sigma}^2\left(\mathbf{X}^T\mathbf{X}\right)^{-1}_{ii}}}$$

And for the p-values, that give us the significance of our statistical results:

$$p_i = P\left(Z_i > |z_i|\right)$$

Now we can also code in order to get all the relevant estimates and statistical tests, given by the `summary`commando below:

```r
summary(a_model)
```

```
## Model Call:
## mylm(formula = wages ~ education, data = SLID)
##
## Coefficients Summary:
##              Estimate  Std_Error         t            p
## (Intercept) 4.9716912 0.53415963  9.307501 1.308757e-20
## education   0.7923091 0.03905069 20.289248 1.600242e-91
##
## F-statistic: [1] 411.4471
## Chi-squared Statistic: [1] 1639617
## Chi-squared p-value: [1] 0
## R-squared: [1] 0.09358627
```

Now we interpret the results. We know that $\mathbf{Y}$ increases by β_i and that we get extremely big t-values, which is normal as we get very small p-values.

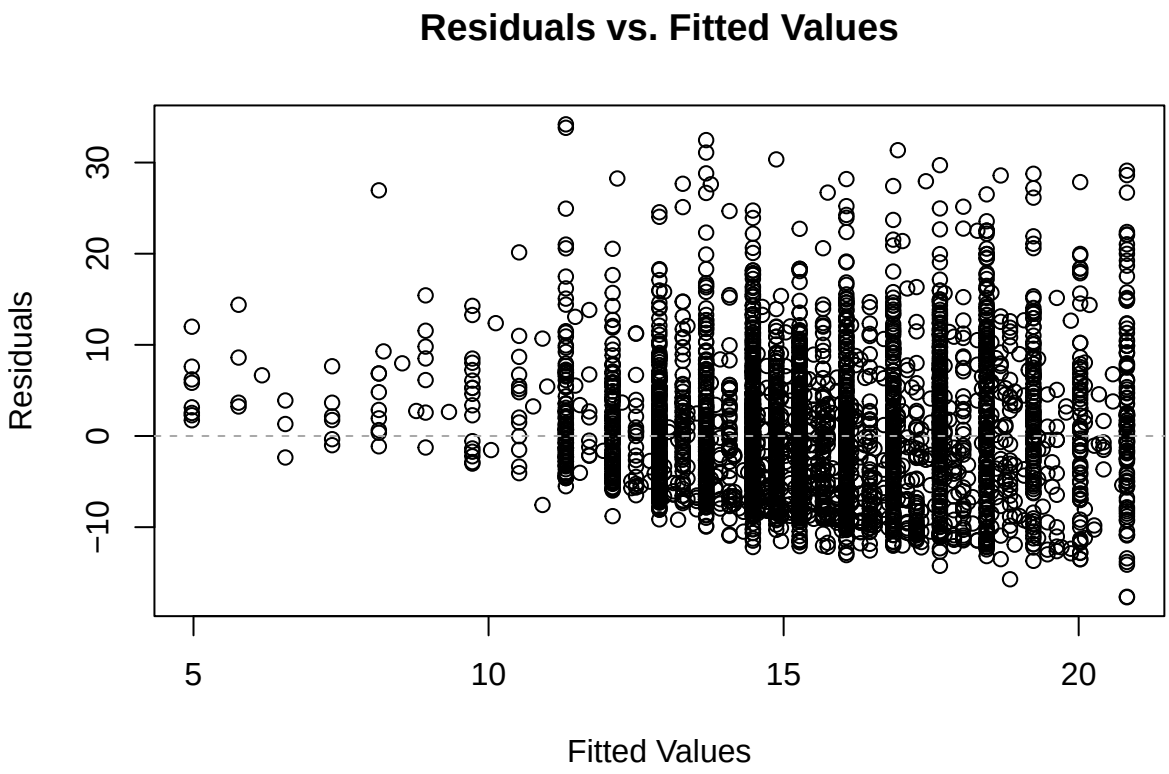
```r
summary(a_model_test)
```

```
##
## Call:
## lm(formula = wages ~ education, data = SLID)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -17.688  -5.822  -1.039   4.148  34.190
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  4.97169    0.53429   9.305   <2e-16 ***
## education    0.79231    0.03906  20.284   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 7.492 on 3985 degrees of freedom
## Multiple R-squared:  0.09359,    Adjusted R-squared:  0.09336
## F-statistic: 411.4 on 1 and 3985 DF,  p-value: < 2.2e-16
```

After comparing `mylm` and `lm` we notice that everything is similar apart from the t-values as we are using a z-test, making it a plausible result.

c)

```r
plot(a_model)
```



This plot shows the predicted values on the x-axis and the residuals on the y-axis. We can see that the average of the residuals is close to zero, which matches our assumption from part 1. However, there's a noticeable tunnel-like pattern, suggesting that the residuals are not independent of the predicted values.

d)

```
a_model$SSE
```

```
## [1] 223694.3
```

```
a_model$SST
```

```
## [1] 246790.5
```

```
a_model$p_chi
```

```
## [1] 0
```

Here we see that the SSE is 223694.3 and the SST is 246790.5. The SST has a χ^2 test that equals to 0 and the degrees of freedom are $n - p = 3985$.

In addition, we want to test the significance of regression using χ^2. For that we need an F-statistic given by:

$$F = \frac{\frac{n-p}{k} \cdot R^2}{1 - R^2}$$

From this, we know that:

$$kF \sim \chi^2_k$$

Which we use to find our p-values. Below, we see that our z-test squared equals to χ^2.

```
a_model$chi_statistic
```

```
## [1] 1639617
```

```
a_model$z_values[2]^2
```

```
## [1] 411.6536
```

e)

R^2 gives the proportion of variance explained by the model and it is given by this:

$$R^2 = 1 - \frac{\text{SSE}}{\text{SST}}$$

In our model it has the value of:

```
a_model$r_squared
```

```
## [1] 0.09358627
```

Part 3: Multiple linear regression

a)

Everything we have shown earlier also applies to multiple linear regression. So we can now we can use our source code on a model with multiple covariates.

```
a_model <- mylm(wages ~ education, data = SLID)
b_model <- mylm(wages ~ education + age, data = SLID)
print(b_model)
```

```
## Function Call Details:
## mylm(formula = wages ~ education + age, data = SLID)
##
## Model Parameters:
##   (Intercept) education       age
## 1   -6.021653 0.9014644 0.2570898
```

b)

```r
summary(b_model)
```

```
## Model Call:
## mylm(formula = wages ~ education + age, data = SLID)
##
## Coefficients Summary:
##               Estimate   Std_Error         t             p
## (Intercept) -6.0216529 0.618690864 -9.732894  2.182914e-22
## education    0.9014644 0.035746370 25.218347 2.520521e-140
## age          0.2570898 0.008947866 28.731967 1.521861e-181
##
## F-statistic: [1] 660.7096
## Chi-squared Statistic: [1] 2632267
## Chi-squared p-value: [1] 0
## R-squared: [1] 0.2490697
```

The coefficients for the multiple linear regression model with `education` and `age` predicting wage were calculated as shown in part 1. The estimates for `education` and `age` are 0.9014644 and 0.2570898, respectively.

Using the method from part 2, we found the z-test for these coefficients, assuming the null hypothesis that they are zero. The summary shows that the p-values for both coefficients, along with the intercept, are very low. This means that `education` and `age` are likely significant predictors of wage.

In simple terms, this means that for each extra year of age, the expected wage goes up by 0.2570898. Similarly, each additional year of education is expected to increase the wage by 0.9014644.

c)

To examine the impact of age and education individually and in combination, we fit three separate models:

- One model using only `education` as a predictor.
- One model using only `age` a predictor.
- One model using both `education`and `age` as predictors.

```r
print(a_model)
```

```
## Function Call Details:
## mylm(formula = wages ~ education, data = SLID)
##
## Model Parameters:
##   (Intercept) education
## 1    4.971691 0.7923091
```

```r
c_model = mylm(wages ~ age, data = SLID)
print(c_model)
```

```
## Function Call Details:
## mylm(formula = wages ~ age, data = SLID)
##
## Model Parameters:
##   (Intercept)       age
## 1    6.890901 0.2331079
```

```r
print(b_model)
```

```
## Function Call Details:
## mylm(formula = wages ~ education + age, data = SLID)
##
## Model Parameters:
##   (Intercept) education       age
## 1   -6.021653 0.9014644 0.2570898
```

The differences in parameter estimates grow because the covariates are not independent of each other. This is reflected in the covariance matrix for the model that includes both `age` and `education`, where we see non-zero off-diagonal elements. These off-diagonal values show that changes in one covariate can impact the other related covariates.

If the covariates were truly independent, adding or removing a covariate would not affect the coefficients of the remaining covariates. However, this isn't the case here. It makes sense, given that as people get older, they often acquire more education, and the reverse can also be true. This interdependence between `age` and `education` is why we see these variations in the parameter estimates.

Part 4: Testing the `mylm` package

In this task we will use three different models in order to evaluate whether the `mylm` packages works, by fitting these and including diagnostics. These three models are:

- A model with `sex`, `age`, `language` and `education`^{2}.
- A model with `language` and `age` with an interaction term between the predictors.
- A model with `education` and no intercept.

For the first model:

```r
model_1 <- mylm(wages ~ sex + age + language + I(education^2), data = SLID)
summary(model_1)
```

```
## Model Call:
## mylm(formula = wages ~ sex + age + language + I(education^2),
##     data = SLID)
##
## Coefficients Summary:
##                    Estimate   Std_Error          t             p
## (Intercept)    -1.87553134 0.440013681 -4.2624387  2.022080e-05
## sexMale         3.40870006 0.208262748 16.3673057  3.273901e-60
## age             0.24862499 0.008656104 28.7225042 1.997899e-181
## languageFrench -0.07553202 0.424815732 -0.1777995  8.588804e-01
## languageOther  -0.13454020 0.322909303 -0.4166501  6.769343e-01
## I(education^2)  0.03481515 0.001288925 27.0110041 1.097508e-160
##
## F-statistic: [1] 344.8469
## Chi-squared Statistic: [1] 1372836
## Chi-squared p-value: [1] 0
## R-squared: [1] 0.3022198
```

First we look at the p-values and notice that for `sex`, `age`, and `education`^{2} affect wages significantly, because of their low p-values, but the language variable isn't as clear since its p-value doesn't show a strong effect.

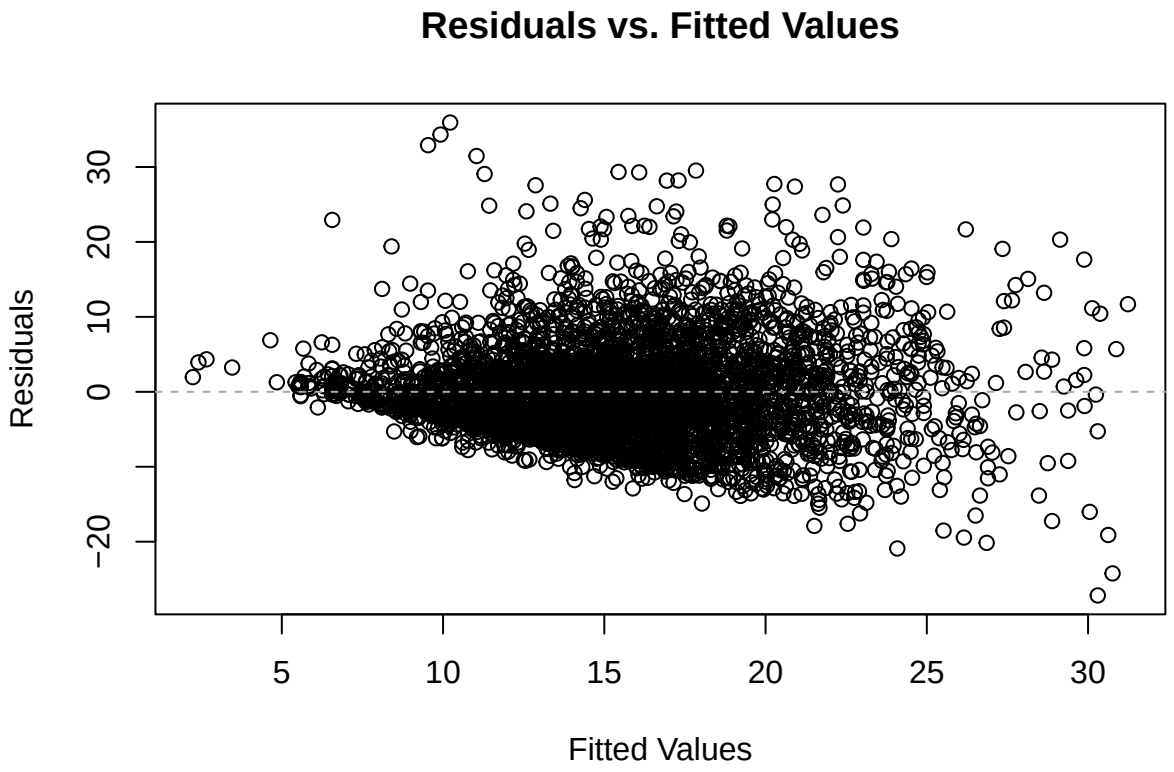
The model also uses a quadratic term for `education`, meaning the relationship with wages isn't just a straight line. Wages grow faster with more `education`, but there's no obvious reason why this quadratic response is better than a linear one.

Looking at the model summary, the negative intercept suggests that with all variables set to zero, the predicted wage is negative. This doesn't seem very logical, which could be a sign that the model isn't fitting the data very well.

Lastly, the "base case" in this model is for a female, English-speaking newborn with no education. Because this situation is far from the actual area of interest, the strange negative intercept might not be a big deal after all.

Now we plot:

```
plot(model_1)
```



Here, we can see a clear pattern where the residuals show a tunnel-like shape. Lower fitted values have lower residuals, while higher fitted values have higher residuals. This suggests that the model might be heteroskedastic and not a great fit.

To make the model better, we could either remove the language variable, since its p-values weren't very clear, or switch to using a linear term for education instead of squaring it, as we've seen that linear education has worked well in the earlier tasks.

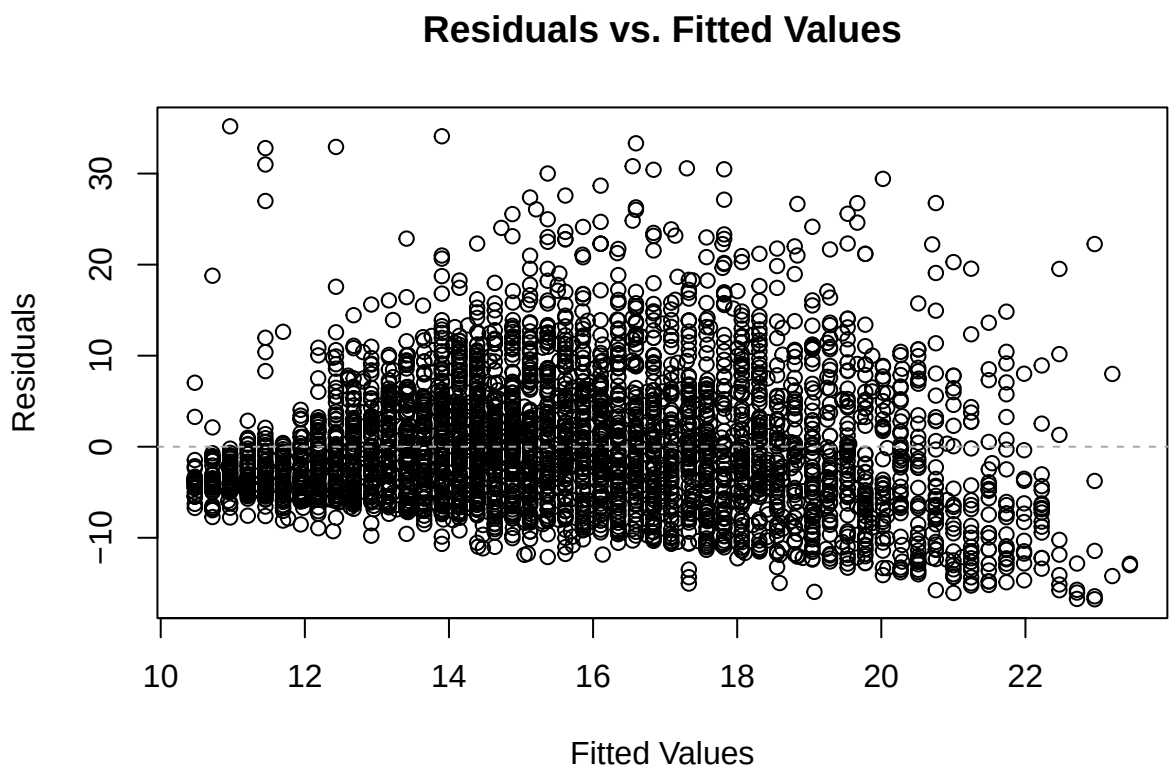
For the second model:

```r
model_2 <- mylm(wages ~ age * language, data = SLID)
summary(model_2)
```

```
## Model Call:
## mylm(formula = wages ~ age * language, data = SLID)
##
## Coefficients Summary:
##                          Estimate  Std_Error          t             p
## (Intercept)          6.55579376 0.41037150 15.9752657  1.900430e-57
## age                  0.24485160 0.01067947 22.9273089 2.482113e-116
## languageFrench       2.86062508 1.59487047  1.7936410  7.287049e-02
## languageOther        0.84862130 1.23425214  0.6875591  4.917305e-01
## age:languageFrench -0.08392752 0.04042557 -2.0760996  3.788474e-02
## age:languageOther  -0.03701381 0.02931813 -1.2624888  2.067730e-01
##
## F-statistic: [1] 120.2053
## Chi-squared Statistic: [1] 478537.3
## Chi-squared p-value: [1] 0
## R-squared: [1] 0.1311705
```

Here, only two coefficients really stand out because they have very low p-values: age and the intercept. The rest don't show clear significance, though `languageFrench` and `age:languageFrench` could be considered significant at the 0.05 level.

```r
plot(model_2)
```



Analyzing this plot we notice that this model is also heteroskedastic as it presents a tunnel-like shape, meaning that the variance increases for bigger fitted values. This means that we have no constant variance which we assumed for the residuals and it is therefore not the best fit.

To make this model better, it makes sense to remove the language variable and replace it with other covariates. Based on the previous parts, language doesn't seem to be one of the most influential predictors.

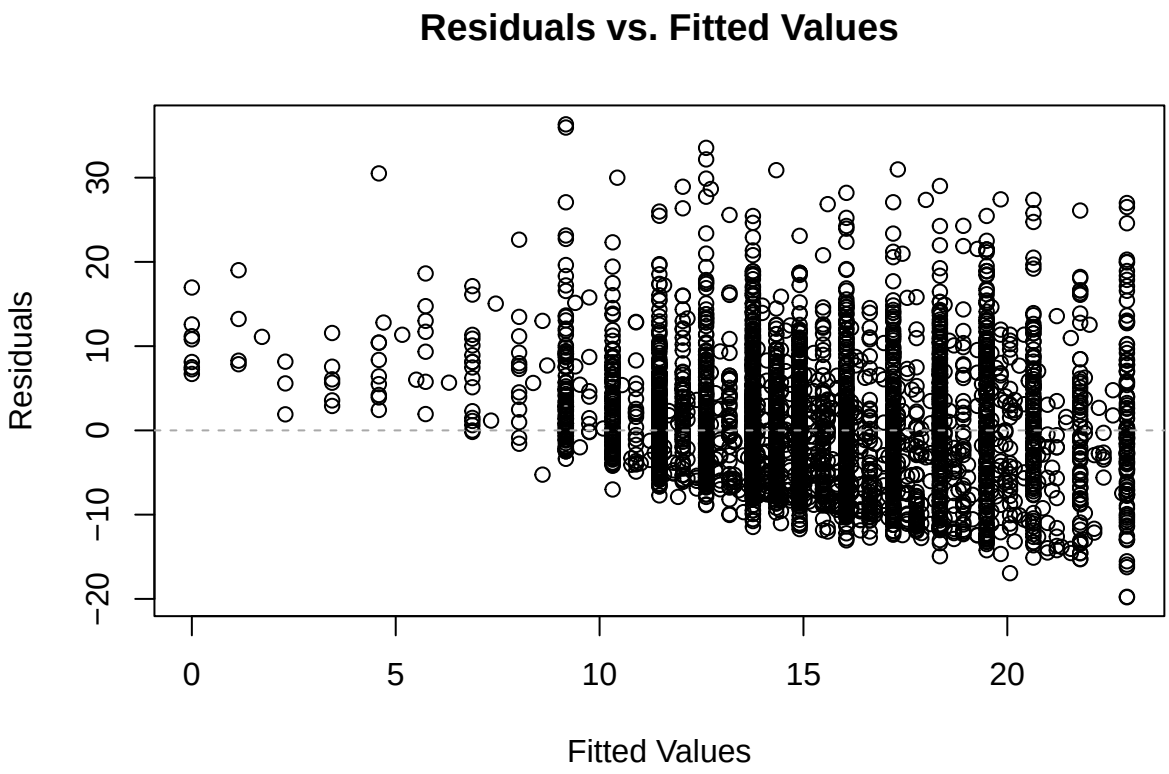
For the third model:

```
model_3 <- mylm(wages ~ -1 + education, data = SLID)
summary(model_3)
```

```
## Model Call:
## mylm(formula = wages ~ -1 + education, data = SLID)
##
## Coefficients Summary:
##           Estimate   Std_Error        t p
## education 1.146697 0.008766101 130.8104 0
##
## F-statistic: [1] Inf
## Chi-squared Statistic: [1] Inf
## Chi-squared p-value: [1] 0
## R-squared: [1] 0.07389171
```

Here, there is only one coefficient, so it's not helpful to display the statistics. With just one coefficient, it's automatically significant.

```
plot(model_3)
```



This plot also reveals a pattern of heteroskedasticity in the residuals, similar to the patterns seen in the first two models. This indicates that the model may not be fitting the data well.

To improve the model, consider including additional covariates, such as the intercept or education, which have proven to be reliable in previous analyses.

Source Code

```
# Select Build, Build and reload to build and lode into the R-session.

mylm <- function(formula, data = list(), contrasts = NULL, ...){
  # Extract model matrix & responses
  mf <- model.frame(formula = formula, data = data)
  X  <- model.matrix(attr(mf, "terms"), data = mf, contrasts.arg = contrasts)
  y  <- model.response(mf)
  terms <- attr(mf, "terms")


  # Add code here to calculate coefficients, residuals, fitted values, etc...
  # and store the results in the list est

  # beta and fitted values
  matrix_inverse <- solve(crossprod(X))
  coefficients <- matrix_inverse %*% t(X) %*% y
```

```r
predicted_values <- X %*% coefficients
n <- length(predicted_values)

#intercept-only model
mean_y <- mean(y)
intercept_only_predictions <- rep(1, length(predicted_values)) * mean_y
residuals <- y - predicted_values
residual_sum_squares <- sum(residuals^2)
total_sum_squares <- sum((y - intercept_only_predictions)^2)
residual_variance <- residual_sum_squares / length(y)
regression_covariance <- residual_variance * solve(crossprod(X))

#z-values and p-values for each coefficient
num_coeffs <- length(coefficients)
t_values <- numeric(num_coeffs)
p_values <- numeric(num_coeffs)
for (index in seq_len(num_coeffs)) {
  standard_error <- sqrt(diag(regression_covariance)[index])
  t_values[index] <- (coefficients[index] - 0) / standard_error

  p_values[index] <- 2 * pnorm(abs(t_values[index]), lower.tail = FALSE)
}

#F-statistic, Chi-squared and R^2
k <- num_coeffs - 1
df <- n - num_coeffs
df_residuals <- length(predicted_values) - length(coefficients)


explained_variance <- total_sum_squares - residual_sum_squares
variance_of_residuals <- residual_sum_squares / df_residuals
f_statistic <- ((total_sum_squares/residual_sum_squares)-1) * ((n - num_coeffs)/k)

chi_squared <- df * f_statistic
p_value_chi <- pchisq(chi_squared, df= (n - num_coeffs), lower.tail=FALSE)

r_squared <- 1 - residual_sum_squares / total_sum_squares

est <- list(
  coefficients = coefficients,                    # Beta values
  p_values = p_values,                             # p-values for coefficients
  z_values = t_values,                            # z-values for coefficients
  covariance_matrix = regression_covariance,            # Covariance matrix of regression coefficient
  fitted_values = predicted_values,              # Fitted values from the model
  residuals = residuals,                         # Residuals of the model
  f_statistic = f_statistic,                      # F-statistic
  SSE = residual_sum_squares,                    # Sum of Squared Errors
  SST = total_sum_squares,                       # Total Sum of Squares
  chi_statistic = chi_squared,                    # Chi-squared statistic
  p_chi = p_value_chi,                           # p-value for the Chi-squared statistic
  r_squared = r_squared,                              # R-squared value
  model = mf                              # The model frame, assuming `mf` refers to the model frame
)
```

```r
  # Store call and formula used
  est$call <- match.call()
  est$formula <- formula

  # Set class name. This is very important!
  class(est) <- 'mylm'

  # Return the object with all results
  return(est)
}

print.mylm <- function(object, ...){
  # Code here is used when print(object) is used on objects of class "mylm"
  # Useful functions include cat, print.default and format
  cat('Function Call Details: \n')
  print(object$call)
  coefficients <- object$coefficients
  transposed_coeffs <- t(coefficients)
  coef_df <- as.data.frame(transposed_coeffs, stringsAsFactors = FALSE)
  colnames(coef_df) <- rownames(coefficients)
  rownames(coef_df) <- NULL
  cat('\nModel Parameters: \n')
  print(coef_df)
}


summary.mylm <- function(object, ...){
  # Code here is used when summary(object) is used on objects of class "mylm"
  # Useful functions include cat, print.default and format
    cat("Model Call: \n")
    print(object$call)
    coef_estimates <- object$coefficients
    std_errors <- sqrt(diag(object$covariance_matrix))
    z_scores <- object$z_values
    p_values <- object$p_values
    coefficient_summary <- data.frame(
      Estimate = coef_estimates,
      Std_Error = std_errors,
      t = z_scores,
      p = p_values
    )
    rownames(coefficient_summary) <- rownames(object$coefficients)
    cat("\nCoefficients Summary: \n")
    print(coefficient_summary)
    cat("\nF-statistic: ")
    print(object$f_statistic)
    cat("Chi-squared Statistic: ")
    print(object$chi_statistic)
    cat("Chi-squared p-value: ")
    print(object$p_chi)
    cat("R-squared: ")
    print(object$r_squared)
}
```

```r
plot.mylm <- function(object, ...){
  # Code here is used when plot(object) is used on objects of class "mylm"
  predicted_vals <- object$fitted_values
  residuals_vals <- object$residuals
  plot(predicted_vals, residuals_vals,
       xlab = "Fitted Values",
       ylab = "Residuals",
       main = "Residuals vs. Fitted Values")
  abline(h = 0, col = "darkgray", lty = 2)
}



# This part is optional! You do not have to implement anova
anova.mylm <- function(object, ...){
  # Code here is used when anova(object) is used on objects of class "mylm"

  # Components to test
  comp <- attr(object$terms, "term.labels")

  # Name of response
  response <- deparse(object$terms[[2]])

  # Fit the sequence of models
  txtFormula <- paste(response, "~", sep = "")
  model <- list()
  for(numComp in 1:length(comp)){
    if(numComp == 1){
      txtFormula <- paste(txtFormula, comp[numComp])
    }
    else{
      txtFormula <- paste(txtFormula, comp[numComp], sep = "+")
    }
    formula <- formula(txtFormula)
    model[[numComp]] <- lm(formula = formula, data = object$model)
  }

  # Print Analysis of Variance Table
  cat('Analysis of Variance Table\n')
  cat(c('Response: ', response, '\n'), sep = '')
  cat('          Df  Sum sq X2 value Pr(>X2)\n')
  for(numComp in 1:length(comp)){
    # Add code to print the line for each model tested
  }

  return(model)

}
```