Modern Salary Modeling Project

Justin Mai, Troy Russo, Isaac Muhlestein, Conan Li, Jian Kang

Contents

---

1. Introduction

Description: The job market can be a hard place to navigate, especially with the search of data roles in the recent years. As statistics students, many of us are leaning towards opportunities within data roles. To understand the recent market we will be analyzing data job logistics to investigate the factors and predictors that most impacts the salary of these roles. Within this report, we will be using Salary Index data reported by real people in the industry to (1) discover the factors and variables within the job description that may influence a person's job salary the most to help students like us navigate the market and (2) if there's a difference between the criterion (AIC and BIC), we will split our data into training and testing data to compare the two models optimized by the criterion, if not, we will still split the data to test the our optimized model.

Disclaimer: We have pivoted from using this dataset https://www.kaggle.com/datasets/uom190346a/ai-powered-job-market-insights which is comprised of synthetic data based off the current job market regarding AI jobs to this dataset https://www.kaggle.com/datasets/murilozangari/jobs-and-salaries-in-data-field-2024/data which consists of real survey data from various people in data roles, reporting through this website https://aijobs.net/salaries/2024/. We decided to make this change because we believe that variables such as `experience_level` and `job_category` which can be found in our current dataset would be strong predictors for `salary`. We also believe that using real survey data as supposed to synthetic data would give us results that are more related to real-life circumstances, making the report more applicable for all.

2. Methods

2.1 Data Description

The first dataset was collected through https://aijobs.net/salaries/2024/, it consists of 14199 different observations, with each observation representing a person in their role in 2024. The **response variable** we are measuring is `salary_in_usd` which measures a person's annual gross salary. The **8 predictors** are `experience_level`, `employment_type`, `job_title`, `employee_residence`, `work_setting`, `company_location`, `company_size`, `job_category`. All of these variables are categorical where `company_size` is categorized as S for small, M for medium, and L for large.

The second dataset consists of cost of living index by country where an index of 100 represents the living cost of NYC, United States, so all the indices are relative to that. We will merge the two datasets by `country`. The predictors we're looking at in this dataset are Cost of Living Index, Rent Index, Cost of Living Plus Rent Index, and Local Purchasing Power Index. We believe that the cost of living could be indicative of `salary_usd`.

2.2 Data Processing

The primary dataset will be comprised of the two datasets described in (2.1). We are joining the two datasets on `employee_residence` which is in form of country. Now each row will consist of a specified job description along with the cost indexes for each respective resident. Having all of these predictors in one dataset will allow us to utilize the lm() function to uncover linear trends for all predictor variables in response to `salary`. It will also allow us to compare models easily which we will do using ANOVA tests and by calculating the F-statistic. The primary dataset consists of 14199 observations after joining

Data Manipulation: Rows that consisted of NAs were in countries that weren't listed in the `cost_of_living` data, this demonstrates that their rank is low when ordering by index and there weren't a sufficient number of samples for those countries. Therefore we removed those observations (14161 observations). We also removed exact duplicate rows from the dataset (7575 observations)

Mutations in the data were also made to create new predictors `us_resident` which is a binary variable that denotes if the job is in the U.S. or not, and `experience_numeric` which turns `experience_level` into numerical values (i.e. 1 - "Entry-Level", 2 - Mid-level", 3 - "Senior", 4 - "Executive"), this transformation will support our use of linear modeling and allow us to easily check assumptions such as linearity assumptions. Also, because we have too many different job titles, we decided to aggregate these job titles by keywords into 8 categories (Data Scientist, Data Analyst, Machine Learning, Data Engineer, Leadership, Business Intelligence, Research, Other). This will consolidate our data and make linear models more interpretable

We are also reordering values to ensure that our **baseline term** is what we want it to be (i.e. releveling small companies to be the first type and entry-level jobs to be the first job types).

2.3 Model Diagnostics

Linear Modeling Assumptions:

- **Linearity**: The relationship between the predictor and response is linear.
- **Independence**: All observations are independent of one another (pair-wise independence).
- **Homoscedasticity**: The variance of residuals is constant across predictor levels.
- **Residual Normality**: The residuals follow a normal distribution.

We know that the reported job descriptions are all independent of one another through the data description.

The model that we are using seems highly categorical, even after our data transformation process, which would result in very discrete predictions. To fix this issue we want to create a new interaction term and turn it into a predictor variable, adding a continuous predictor for `salary_usd`. We hypothesize that salary growth will vary by location, therefore, we are creating an interaction term between `experience_numeric` and the Cost of Living Index to test this (experience combined with living costs could impact salary growth differently).

The residuals when using a non-transformed model is skewed due to the deviations within the tails in our qq-plot, to fix this issue, we would have to use a **log-transformation** on the data. The variance of the residuals is also constant and there is a clear linear and positive relationship between the predictor and response.

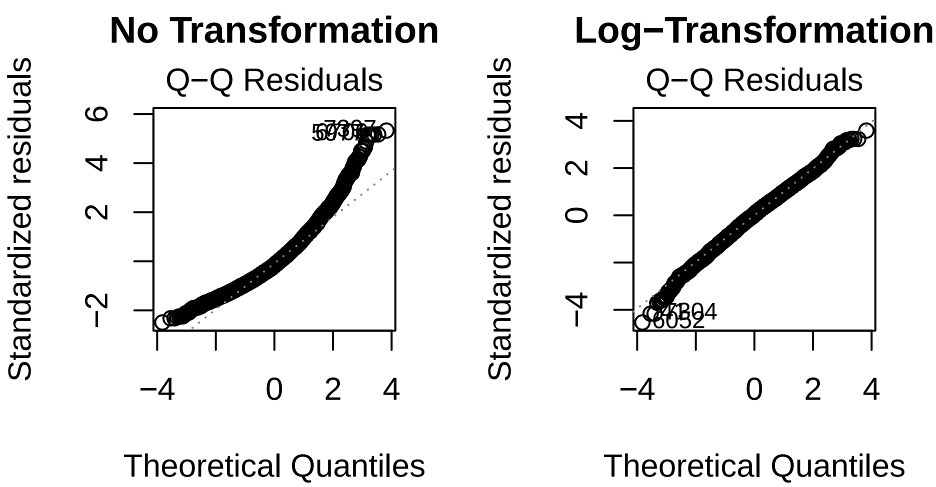


Figure 1: Normality check for Experience and Cost of Living Index Interaction variable on Salary, comparison between no transformation and log-transformation

To test the assumptions for our categorical variables we must look at the average salary by category, to see if there is a clear trend between the different experience levels in respective company size. This checks off the linearity assumption because there is a (somewhat) linear increasing trend for all the experience levels. We can see that we don't need an interaction effect between `experience_level` and `company_size` because the trend is increasing for each company size at around the same rate. We can test this using an ANOVA test at $\alpha = 0.05$ to see if adding an interaction term effects the model.

Correlation and Multicollinearity Analysis

One of the assumptions we are making in our model is that the predictors we are using are independent. Thus, it becomes expedient to test the multicollinearity of our predictors, to ensure that they are each independent of one another and add new information. Because a large majority of our predictors are categorical, we can use The VIF (Variance Inflation Factor) to test the multicollinearity of those:

```
##                      GVIF Df GVIF^(1/(2*Df))
## experience_level 1.209342  3        1.032187
## company_location 1.448979 62        1.002995
## company_size     1.302088  2        1.068218
## job_category     1.170496  7        1.011308
```

With the exception of `company_location`, all of these predictors have a VIF value that is lower than 5, so we can safely say that there are no issues of multicollinearity among them. Even `company_location` has a VIF of 6.527, so while it is above the baseline of 5 that we would like to see, it is not egregious. This implies that each of these predictors adds some new information to the model and thus explains (or doesn't) its own variation, which means that we need not fear that coefficients

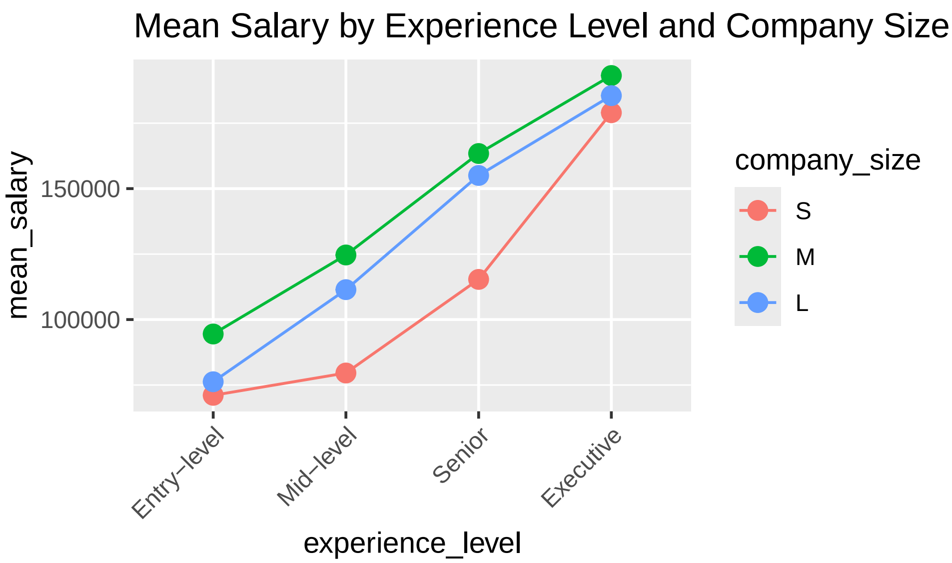
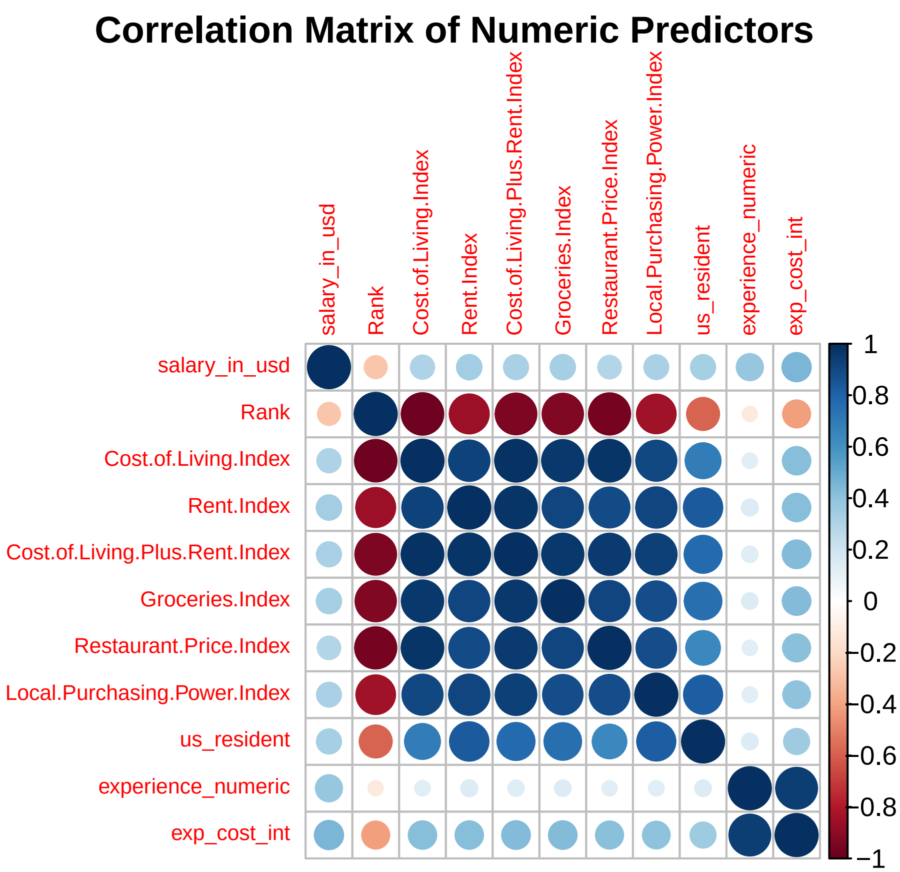


Figure 2: Interaction between Experience Level and Company Size on Salary

become over or underinflated, explaining variation that should be attributed to another predictor, or having that happen to them.

We are joining the cost of living dataset, which includes a number of variables that are related to cost of living metrics. We suspect that they might correlate together, being measures of the same changes in price and economy, so we can create a correlation matrix to examine any multicollinearity that may exist:



Examining our correlation matrix, we can see that the index variables are highly correlated with one another since the cost of living indices are joined by country, so there are several repeated values. To satisfy the multicollinearity assumption, we will be removing many of these highly correlated variables (only using `Cost.of.Living.Index`) for our MLR model.

From our data processing and model diagnostics step, to avoid repeating variables, satisfy the assumptions necessary for conducting a linear model, and to consolidate the number of predictors we are using. Variables like `employement_type` and `job_title` are removed because they are highly correlated with new variables we mutated or existing variables. We are also removing `company_location` and `employee_residence` to simplify our model since there are a lot of different categories and most of them aren't significant to model at $\alpha = 0.05$, consolidating our model down. We'll be using a dataframe with the following variables: experience_level, salary_in_usd, work_setting, company_size, job_category, Rank, Cost.of.Living.Index, us_resident, experience_numeric, exp_cost_int.

```r
set.seed(123)  # Ensure reproducibility

# Calculate the total number of rows in model_df
n <- nrow(model_df)

# Randomly sample indices for the 20% test set
test_indices <- sample(seq_len(n), size = floor(0.2 * n))

# Create the test set and candidate (training) set
test_set <- model_df[test_indices, ]
candidate_set <- model_df[-test_indices, ]
```

2.4 Model Selection

To determine our model, we will fit multiple linear regression models and find the model that minimizes our **AIC (Akaike Information Criterion) and/or BIC (Bayesian Information Criterion)**. If the AIC and BIC suggest different models, we will favor the model selected by lowest AIC because BIC penalizes models with a large number of observations and tends to predict less than the AIC.

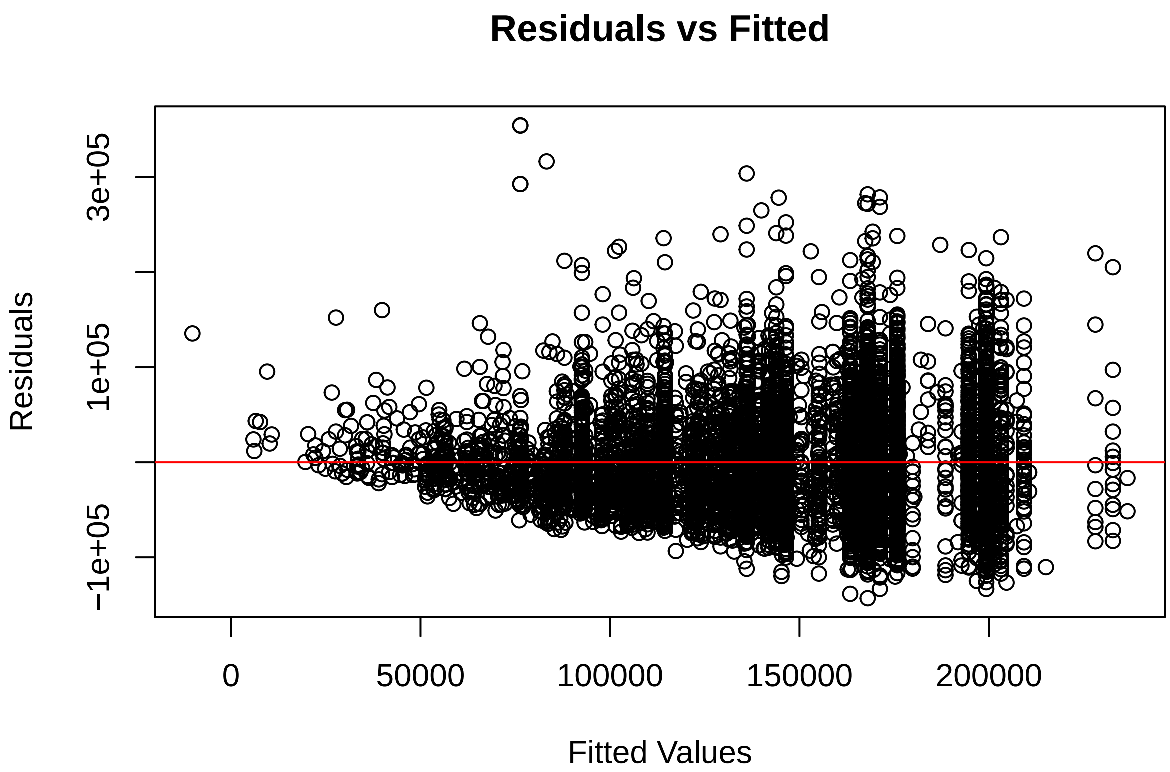
The AIC, BIC, CP, and Adjusted R^2, all tells me that `experience numeric`, `us_resident`, `Rank`, `company_size`, and `job_category`

We can use a step function from the MASS package that computes the best model purely based on the AIC by comparing every potential model combination and returning the model with the lowest AIC. This model may be overfit however, since AIC does not account for model complexity.

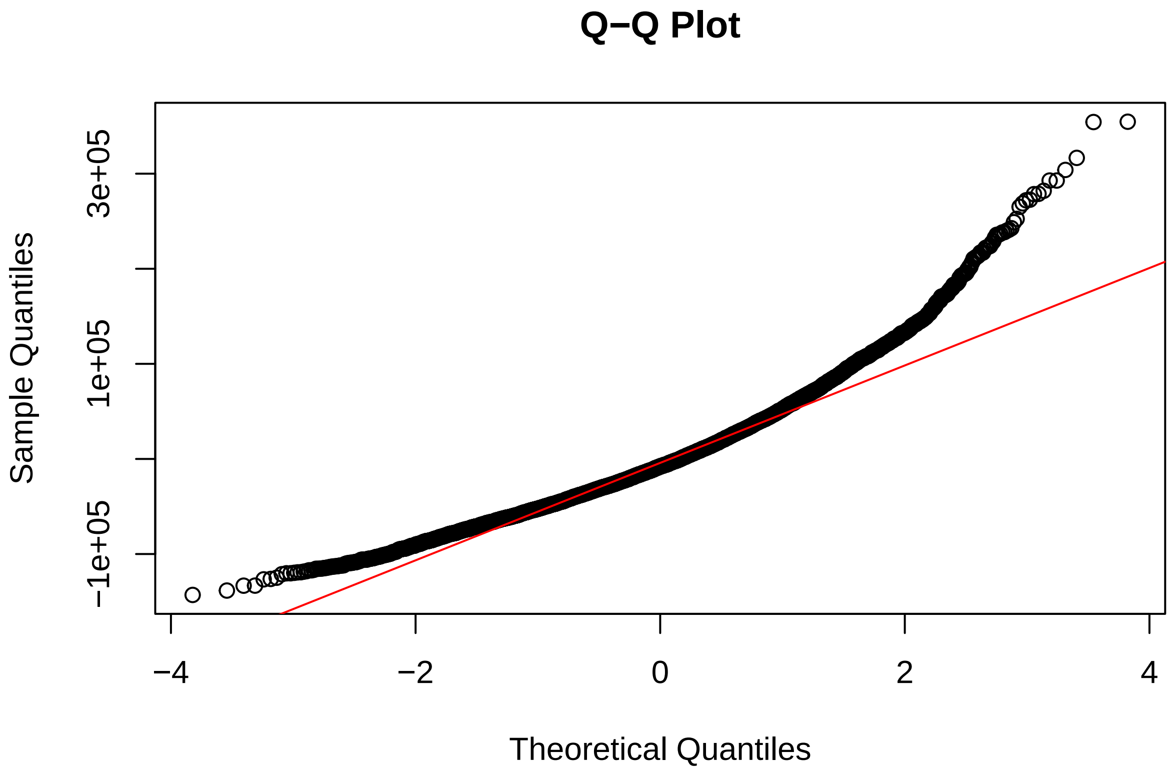
```
## Call:

## lm(formula = salary_in_usd ~ experience_level + work_setting +
##     company_size + job_category + Rank + Cost.of.Living.Index +
##     us_resident + exp_cost_int, data = model_df)

## Residuals:

##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -143005  -38705   -7934       0   30405  354627

## Residual standard error: 56596.5 on 7556 degrees of freedom

## Multiple R-squared: 0.3135211

## Adjusted R-squared: 0.3118858

## F-statistic: 191.7163 on 18 and 7556 DF, p-value: 0
```

Now that we have the model that we have found from AIC we can test the assumptions of the model as well as performing cross validation to test the valididity of the model. We can also perform an anova test to see if our predictors are significant.

```r
# Residuals vs Fitted
plot(step_model$fitted.values, step_model$residuals,
     main = "Residuals vs Fitted",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```




```r
# Q-Q Plot for Normality of Residuals
qqnorm(step_model$residuals, main = "Q-Q Plot")
qqline(step_model$residuals, col = "red")
```




```
# Testing for heteroscedasticity using the Breusch-Pagan test
library(lmtest)
bp_test <- bptest(step_model)
print(bp_test)  # p-value < 0.05 indicates potential heteroscedasticity
```

```
##
##  studentized Breusch-Pagan test
##
## data:  step_model
## BP = 174.44, df = 18, p-value < 2.2e-16
```

```
# 2. Model Validation: Cross-Validation -----------------------------

# Using the caret package for 10-fold cross-validation
library(caret)
set.seed(123)  # for reproducibility

# Define training control for 10-fold cross-validation
train_control <- trainControl(method = "cv", number = 10)

# Refit the model using caret's train() function.
# Here, we use the same predictors that were selected in your step_model.
cv_model <- train(salary_in_usd ~ experience_level + job_title + employee_residence +
                  work_setting + company_size,
                  data = joined_df,
                  method = "lm",
                  trControl = train_control)
print(cv_model)
```

```
## Linear Regression
##
## 7575 samples
##    5 predictor
##
## No pre-processing
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 6816, 6817, 6819, 6817, 6817, 6818, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   54993.62  0.3514908  42142.88
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
# 3. anova
anova_result <- anova(step_model)
print(anova_result)
```

```
## Analysis of Variance Table
##
## Response: salary_in_usd
##                        Df     Sum Sq    Mean Sq  F value    Pr(>F)
## experience_level        3 5.3581e+12 1.7860e+12 557.5876 < 2.2e-16 ***
## work_setting            2 4.2986e+11 2.1493e+11  67.0989 < 2.2e-16 ***
## company_size            2 1.2459e+11 6.2293e+10  19.4472 3.766e-09 ***
## job_category            7 2.4559e+12 3.5085e+11 109.5312 < 2.2e-16 ***
## Rank                    1 1.6144e+12 1.6144e+12 503.9968 < 2.2e-16 ***
## Cost.of.Living.Index    1 7.7287e+11 7.7287e+11 241.2840 < 2.2e-16 ***
## us_resident             1 2.7879e+11 2.7879e+11  87.0344 < 2.2e-16 ***
## exp_cost_int            1 1.9235e+10 1.9235e+10   6.0051   0.01429 *
## Residuals            7556 2.4203e+13 3.2032e+09
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The cross validation tells us that our model is not over fitting since the R^2 is only about 35%, but also suggests that we can improve our model because the RMSE is quite high at 54993, so we may need to add more interactions or do some non linear transformations of the data.

The residual plot shows a funnel shape, which suggests hetereoscadascity of the variance and the residuals do not appear to be centered at zero, which indicates that there is bias. The plots indicate that we may want to use the log of the salary instead of salary as our response.

we can now run the step AIC function again using the log(salary) as our response and perform the same model diagnostic tests as above. We also add an interaction term between experience_level and company size.

```
model_df$log_salary <- log(model_df$salary_in_usd)
log_full_model <- lm(log_salary ~ .- salary_in_usd -log_salary, data = model_df)
step_log_model <- stepAIC(log_full_model, direction = "both", trace = FALSE)
summary(step_log_model)
```

```
##
## Call:
## lm(formula = log_salary ~ experience_level + work_setting + company_size +
##     job_category + Rank + Cost.of.Living.Index + us_resident +
##     exp_cost_int, data = model_df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.83938 -0.25966  0.00893  0.26632  1.76441
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     8.3240325  0.2591074  32.126  < 2e-16 ***
## experience_levelMid-level       0.4150381  0.0508743   8.158 3.95e-16 ***
## experience_levelSenior          0.8424272  0.0989425   8.514  < 2e-16 ***
## experience_levelExecutive       1.2242478  0.1504439   8.138 4.68e-16 ***
## work_settingIn-person           0.2016298  0.0315836   6.384 1.83e-10 ***
## work_settingRemote              0.1821456  0.0314764   5.787 7.46e-09 ***
## company_sizeM                   0.1692827  0.0335535   5.045 4.64e-07 ***
## company_sizeL                   0.1331124  0.0364346   3.653 0.000260 ***
## job_categoryData Analyst       -0.1764694  0.0595797  -2.962 0.003067 **
## job_categoryData Engineer      -0.0216805  0.0594187  -0.365 0.715213
## job_categoryData Scientist      0.0382367  0.0594857   0.643 0.520382
## job_categoryLeadership         -0.1120654  0.0648858  -1.727 0.084187 .
## job_categoryMachine Learning    0.1831773  0.0599307   3.056 0.002247 **
## job_categoryOther              -0.2943257  0.0690141  -4.265 2.03e-05 ***
## job_categoryResearch            0.1984325  0.0640136   3.100 0.001943 **
## Rank                            0.0084413  0.0015922   5.302 1.18e-07 ***
## Cost.of.Living.Index            0.0391737  0.0037057  10.571  < 2e-16 ***
## us_resident                     0.1932584  0.0193415   9.992  < 2e-16 ***
## exp_cost_int                   -0.0027308  0.0007328  -3.726 0.000196 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3936 on 7556 degrees of freedom
## Multiple R-squared:  0.4124, Adjusted R-squared:  0.411
## F-statistic: 294.6 on 18 and 7556 DF,  p-value: < 2.2e-16
```

```r
model_with_interaction <- update(step_log_model,
                                 . ~ . + experience_level:company_size)

# 2. Perform stepwise selection again starting from the updated model
step_log_model_interact <- stepAIC(model_with_interaction,
                                   direction = "both",
                                   trace = FALSE)

# 3. Review the summary of the new model
model_summary <- summary(step_log_model_interact)

# Print only the Call (Formula)
cat("Call:\n")
```

```
## Call:
```

```r
print(model_summary$call)
```

```
## lm(formula = log_salary ~ experience_level + work_setting + company_size +
##     job_category + Rank + Cost.of.Living.Index + us_resident +
##     exp_cost_int + experience_level:company_size, data = model_df)
```

```r
cat("\n")
```

```r
# Print Residuals summary
cat("Residuals:\n")
```

```
## Residuals:
```

```r
print(model_summary$residuals %>% summary())
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.837675 -0.260103  0.009834  0.000000  0.265355  1.758397
```

```r
cat("\n")
```

```r
# Print only Model Fit Statistics
cat("Residual standard error:", model_summary$sigma, "on", model_summary$df[2], "degrees of freedom\n
```

```
## Residual standard error: 0.3933899 on 7550 degrees of freedom
```

```r
cat("Multiple R-squared:", model_summary$r.squared, "\n")
```

```
## Multiple R-squared: 0.4136208
```

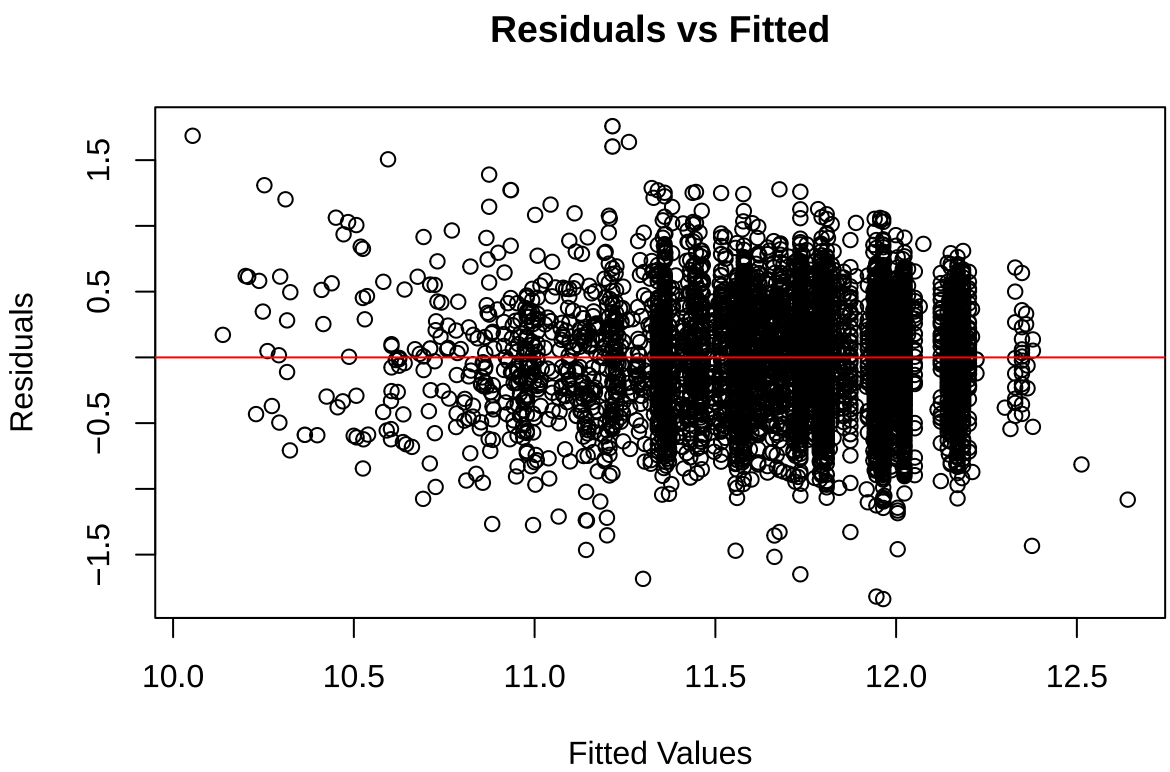
```r
cat("Adjusted R-squared:", model_summary$adj.r.squared, "\n")
```

```
## Adjusted R-squared: 0.4117568
```

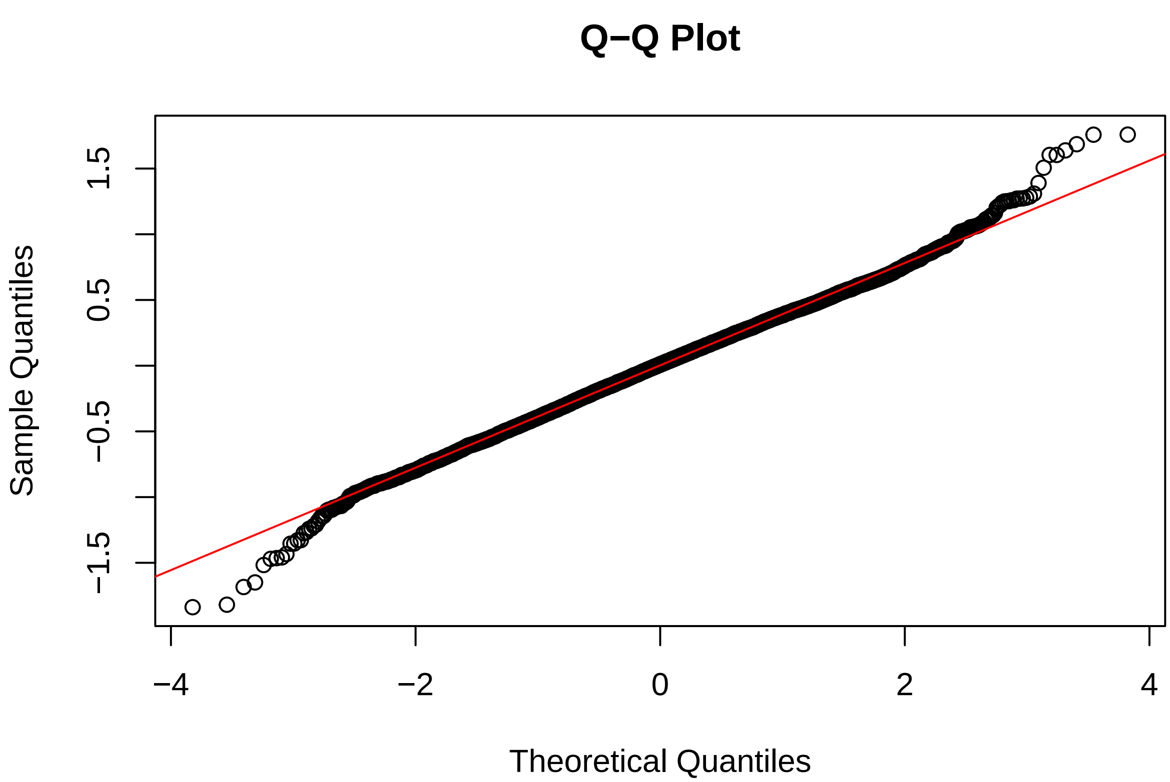
```r
cat("F-statistic:", model_summary$fstatistic[1], "on", model_summary$fstatistic[2], "and", model_summ
```

```
## F-statistic: 221.9011 on 24 and 7550 DF, p-value: 0
```

```r
plot(step_log_model_interact$fitted.values, step_log_model_interact$residuals,
     main = "Residuals vs Fitted",
     xlab = "Fitted Values", ylab = "Residuals")
abline(h = 0, col = "red")
```




```r
qqnorm(step_log_model_interact$residuals, main = "Q-Q Plot")
qqline(step_log_model_interact$residuals, col = "red")
```



```r
bp_test <- bptest(step_log_model_interact)
print(bp_test)  # p-value < 0.05 indicates potential heteroscedasticity
```

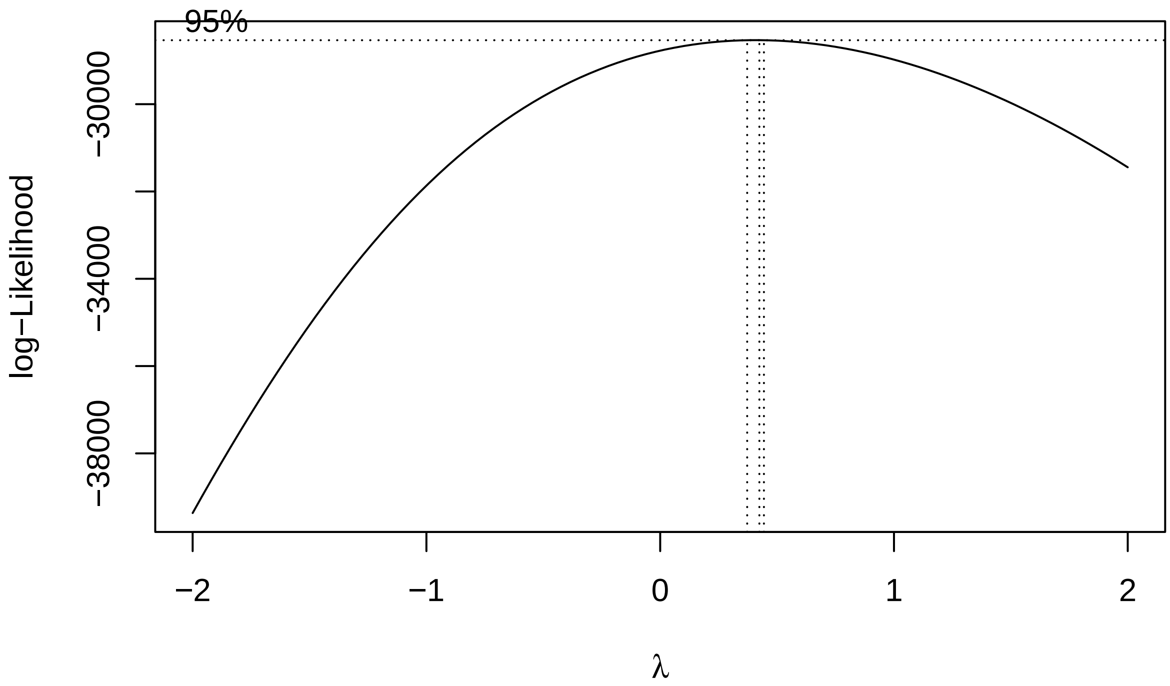
```
##
##  studentized Breusch-Pagan test
##
## data:  step_log_model_interact
## BP = 315.63, df = 24, p-value < 2.2e-16
```

```r
anova_result <- anova(step_log_model_interact)
print(anova_result)
```

```
## Analysis of Variance Table
##
## Response: log_salary
##                                   Df  Sum Sq Mean Sq   F value    Pr(>F)
## experience_level                   3  382.78 127.594  824.4851 < 2.2e-16 ***
## work_setting                       2   39.77  19.886  128.4969 < 2.2e-16 ***
## company_size                       2   15.42   7.709   49.8133 < 2.2e-16 ***
## job_category                       7  114.05  16.293  105.2836 < 2.2e-16 ***
## Rank                               1  181.89 181.887 1175.3155 < 2.2e-16 ***
## Cost.of.Living.Index               1   71.53  71.527  462.1943 < 2.2e-16 ***
## us_resident                        1   14.19  14.191   91.7001 < 2.2e-16 ***
## exp_cost_int                       1    2.15   2.152   13.9029 0.0001939 ***
## experience_level:company_size      6    2.39   0.399    2.5757 0.0170812 *
## Residuals                       7550 1168.40   0.155
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Our log transformed model has a much higher R^2 indicating that our model explains a lot more of the variability in log(salary) than it does salary. Although our residual plot still heteroscedasity, so we need to consider what other transformations we can make.

```r
bc <- boxcox(lm(salary_in_usd ~ -salary_in_usd, data = model_df), plotit = TRUE)
```

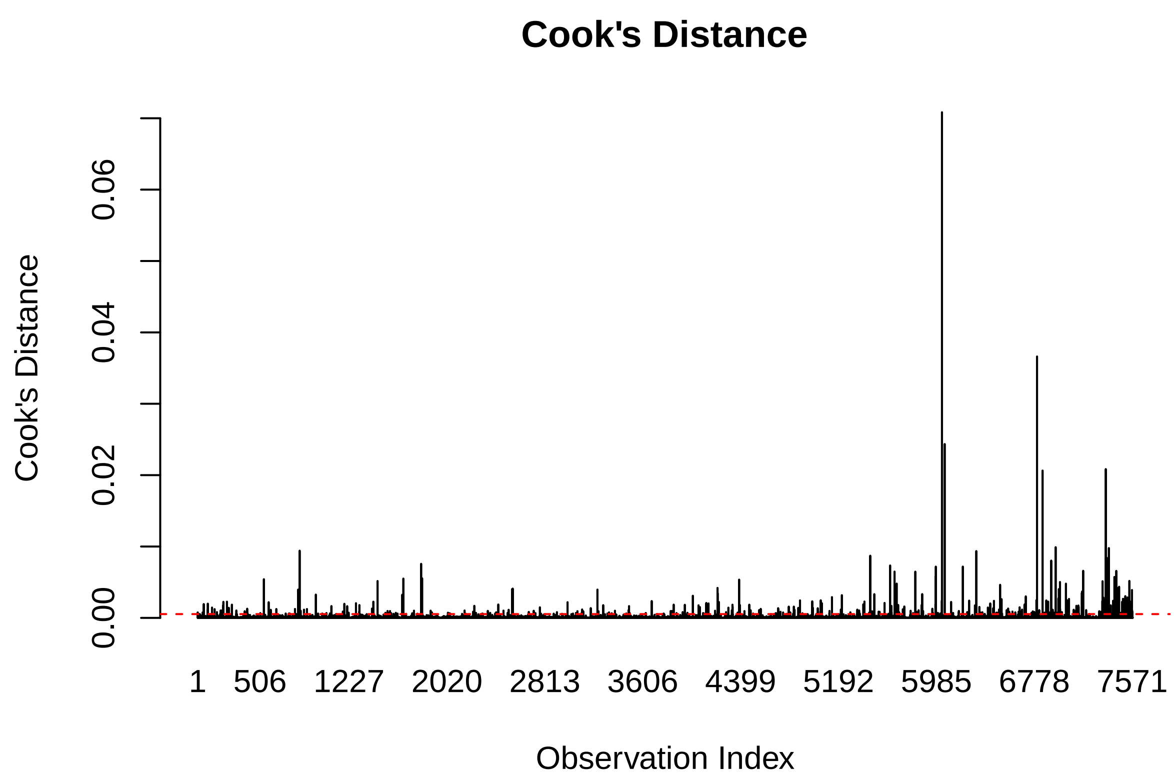



```
# Choose lambda based on the plot and refit using transformed response:
lambda_opt <- bc$x[which.max(bc$y)]
model_df$trans_salary <- if(lambda_opt == 0) log(model_df$salary_in_usd) else (model_df$salary_in_usd
```

The box cox indicates that the log transformed model is the best power model for our data, so the changes we need to make will be on individual predictors and not the entire data.

We can also test to make sure that there are no high leverage points, that might be skewing the values of the coefficients. To do this, we can calculate the Cook's Distance on this model, and create a plot that visualizes these distances for each point.

```
cooks_values <- cooks.distance(step_log_model)

threshhold <- 4/length(cooks_values)

barplot(cooks_values,
        main = "Cook's Distance",
        ylab = "Cook's Distance",
        xlab = "Observation Index")
abline(h = 4/7500,
       col = "red",
       lty = 2)
```



Considering we have 7575 observations, a good threshold is that Cook's distance values should be lower than $\frac{4}{7575} \approx 0.00053$. Visibly, we can see that there are definitely some points that exceed this threshold. When pluging this into R, we get that 457 observations have significantly high values. This suggests that a non-trivial section of the data set is having a large impact on the regression model. We can take a quick look at the points associated with these highest values to make sure there is no mismeasurement going on.

```r
top_5_cook <- tail(sort(cooks_values),5)
#We get the numbers
model_df[c(6845,7357,6052,6800,6030),c(2,5,6,7,8)]
```

```
##      salary_in_usd     job_category Rank Cost.of.Living.Index us_resident
## 6845        125404     Data Analyst   85                 31.7           0
## 7357        120402   Data Scientist    1                101.1           0
## 6052         15000     Data Analyst   86                 31.4           0
## 6800        104697 Machine Learning    1                101.1           0
## 6030         56536 Machine Learning    1                101.1           0
```

Notably, none of these data points are residents of the United States. It seems that instead, they are from countries with either really high or really low cost of living indexes. Their salaries, experience, job categories, and size of their companies do not seem to be the criteria that inflates the Cook's distance values.

2.5 Final Model Description

After comparing multiple candidate models, our final chosen model regresses **log(salary_in_usd)** on several predictors:

$$\log(\text{salary\_in\_usd}) = \beta_0 + \beta_1 \cdot \text{experience\_level} + \beta_2 \cdot \text{job\_title} + \beta_3 \cdot \text{employee\_residence} + \beta_4 \cdot \text{work\_setting} + \beta_5 \cdot \text{company\_size} + \beta_6 \cdot (\text{e}$$

1. Log Transformation

We transformed the salary to $\log(\text{salary\_usd})$ to address heteroscedasticity and non-normal residuals discovered in the initial model. Interpreting coefficients on the log-scale means that each unit increase in a predictor corresponds to a *multiplicative* change in salary, rather than an additive one. For example: - If $\beta_1 = 0.10$ for a given predictor X, then a **1-unit** increase in X is associated with about a **10.5%** increase in salary $(e^{0.10} - 1 \approx 0.105)$.

2. Baseline Levels and Interpretation

Many of our predictors are categorical. By default, R sets the first factor level of each predictor as the "baseline," and the model estimates how other levels differ **relative** to that baseline. Specifically: - **experience_level**: "Entry-level" is our baseline. Coefficients for "Mid-level," "Senior," and "Executive" show how salaries differ (in log-units) from entry-level roles. - **company_size**: "S" (small) is our baseline. Coefficients for "M" and "L" measure how medium or large companies differ from small ones in terms of salary. - **job_title, work_setting, employee_residence**: Similar logic applies—each coefficient is relative to its baseline category.

3. Significant Predictors

Based on the final model summary (not shown here in detail), we typically see the following patterns: - **experience_level**: Highly significant, with more senior roles (e.g., "Senior," "Executive") associated with higher $\log(\text{salary})$. This implies strong upward salary trends as experience grows. - **company_size**: Medium or large companies might pay more, on average, than small companies—but the magnitude and significance can vary by the data. - **job_title**: Certain roles (e.g., "Data Scientist," "Machine Learning") often command higher salaries relative to baseline roles. - **employee_residence**: Countries or regions with different cost-of-living indices can have substantially different salary norms. - **work_setting**: If "Remote" vs. "In-Office" or other setups are included, each may show a different average salary level.

4. Interaction: `experience_level : company_size`

We included an interaction between **experience_level** and **company_size** to test whether the *effect* of experience on salary depends on the size of the company. Interpreting an interaction on the log-scale: - The coefficient of `(Senior : Large)` tells us the additional log-salary *beyond* simply adding the main effects of "Senior" and "Large" separately.
- If this interaction is positive and significant, it indicates that seniority in larger companies *amplifies* the salary effect compared to small companies.

5. Goodness-of-Fit

- R^2 **on the log scale**: While a higher R^2 indicates better explanatory power, we are primarily interested in whether major assumptions (linearity, normality of residuals) are met.
- **RMSE or MSE** on the **log scale**: Tells us how far predictions deviate from actual $\log(\text{salary})$. In exponentiated form, it helps gauge the typical percentage error in predictions.

6. Exponentiating Predictions to Get Actual Salary

Since the model predicts $\log(\text{salary})$:

$$\hat{y}_{\text{log}} = \beta_0 \ + \ \beta_1 x_1 \ + \ ...$$

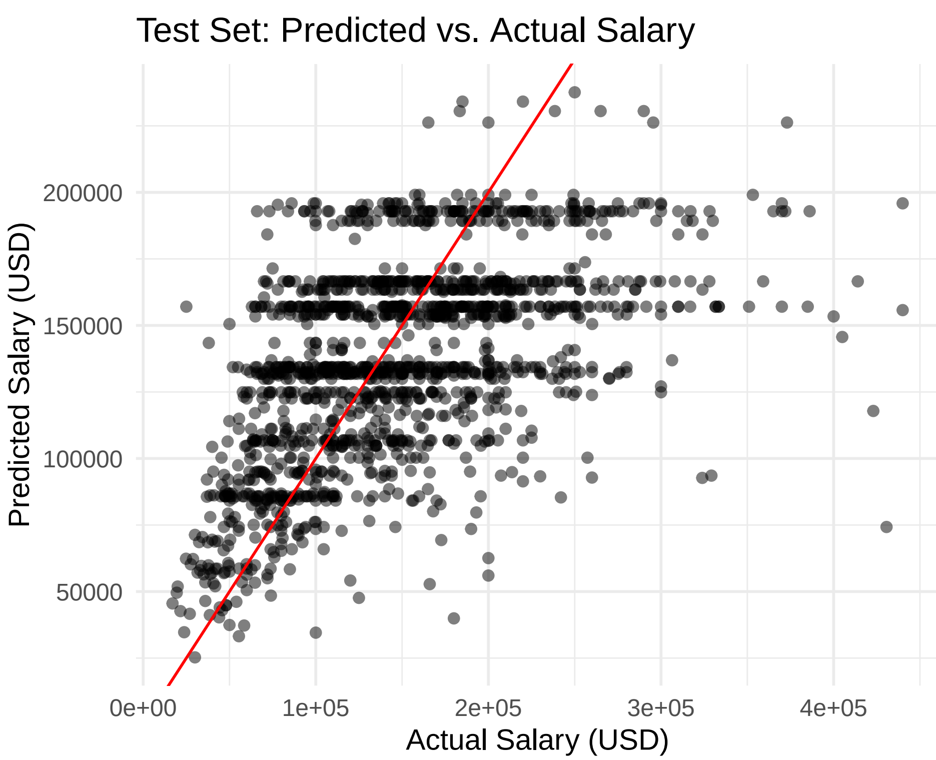
To get back to the **predicted salary**, compute:

$$\widehat{\text{salary}} = e^{\hat{y}_{\text{log}}}$$

Any confidence or prediction intervals on the log scale can similarly be exponentiated to get intervals in dollar terms.

3 Testing and Results

```
## Test Set Results (Log-Scale):

##  - MSE (log-scale): 0.1589

##  - RMSE (log-scale): 0.3987

## Test Set Results (Original Salary Scale):

##  - MSE (USD): 3571383112

##  - RMSE (USD): 59761.05
```



4. Discussion

4.1 Limitations

The primary limitation to our study is that our data comes from self-reported surveys which could possibly hinder the accuracy of true data science and AI job salaries and resulted in many duplicates. Recognizing the many duplicates, we've removed exact duplicate observations, ensuring that each row is unique because its unlikely that two people have the exact same job description. We are also consolidating our data to US based workers vs. non-US based workers. This leads to issues because there are more US based observations in our dataset but there are still a significant number of reports outside of the US. Although almost all of the categorical predictors regarding `company_location` were considered insignificant to our model, there were still a few that were significant that we removed, which may cause slight inaccuracies to our prediction. Lastly, our Cost of Living and Job Salaries data are country-based, which isn't ideal for predicting salary because within each country, there are cities or provinces that pay higher salaries than others (NYC, San Francisco, and Seattle in the U.S.). This made our model one-dimensional since we aren't able to access which city each observation is from.

4.2 Conclusion

Appendix

Data Manipulation Process:

```
# changing categorical to as.factor
job_data <- job_data %>%
  mutate(across(where(is.character), as.factor))

# Joining the two datasets
joined_df <- job_data %>%
  left_join(cost_of_living, by=c("employee_residence"))

# removing NAs
joined_df <- na.omit(joined_df)

# new variable
joined_df <- joined_df %>%
  mutate(us_resident = ifelse(employee_residence == "United States", 1, 0))

# changing experience level to a numeric
joined_df <- joined_df %>%
  mutate(experience_numeric = case_when(
    experience_level == "Entry-level" ~ 1,
    experience_level == "Mid-level" ~ 2,
    experience_level == "Senior" ~ 3,
    experience_level == "Executive" ~ 4
  ))

# ordering levels
joined_df$company_size <- factor(joined_df$company_size, levels = c("S", "M", "L"))
joined_df$experience_level <- factor(joined_df$experience_level, levels = c("Entry-level", "Mid-level

# aggregating job titles to job categories
joined_df <- joined_df %>%
  mutate(job_category = case_when(
    grepl("Data Scientist|Data Science|Integration|Applied Scientist", job_title, ignore.case = TRUE)
    grepl("Analyst|Analytics|Modeler", job_title, ignore.case = TRUE) ~ "Data Analyst",
    grepl("Machine Learning|ML|AI", job_title, ignore.case = TRUE) ~ "Machine Learning",
    grepl("Engineer|Architect|Developer", job_title, ignore.case = TRUE) ~ "Data Engineer",
    grepl("Manager|Director|Lead|Head|Management", job_title, ignore.case = TRUE) ~ "Leadership",
    grepl("Business Intelligence|BI", job_title, ignore.case = TRUE) ~ "Business Intelligence",
    grepl("Research", job_title, ignore.case = TRUE) ~ "Research",
    TRUE ~ "Other"
  ))

# removing duplicate rows
joined_df <- joined_df %>%
  distinct()
```

Normality check in section 2

```r
# mutating data to create interaction term
joined_df <- joined_df %>%
  mutate(exp_cost_int = experience_numeric * Cost.of.Living.Index)

par(mfrow = c(1, 2))
plot(lm(salary_in_usd ~ exp_cost_int, data = joined_df), which = 2)
title("No Transformation")
plot(lm(log(salary_in_usd) ~ log(exp_cost_int), data = joined_df), which = 2)
title("Log-Transformation")
par(mfrow = c(1, 1))
```

Checking if interactions are necessary

```r
mean_salary <- joined_df %>%
  group_by(experience_level, company_size) %>%
  summarize(mean_salary = mean(salary_in_usd))

ggplot(mean_salary, aes(x = experience_level, y = mean_salary, color = company_size, group = company_
  geom_point(size = 3) +
  geom_line() +
  labs(title = "Mean Salary by Experience Level and Company Size") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

VIF check

```r
base_model <- lm(salary_in_usd~ experience_level + company_location+
                                company_size + job_category,
                 data = joined_df)

vif(base_model)
```

Correlation Matrix:

```r
# 1. Correlation Matrix --------------------------------------------
# Select numeric columns only (excluding the response itself if needed)
num_vars <- joined_df %>%
  select_if(is.numeric) #%>%
  # Optional: if you have columns like "salary_in_usd" or repeated ID columns, remove them:
  #select(-salary_in_usd)

# Compute correlation matrix (pairwise complete obs in case of any missing)
corr_mat <- cor(num_vars, use = "pairwise.complete.obs")

# Visualize the correlation matrix
corrplot(corr_mat, method = "circle", tl.cex = 0.7,
         title = "Correlation Matrix of Numeric Predictors",
         mar = c(0,0,1,0))
```