Factors Impacting Number of LinkedIn Job Applications per Post View

JRLK - Jessie Ringness, Rebekah Kim, Laura Cai, Karen Dong

2023-11-14

```
linkedin_subset <- linkedin_subset |>
  filter(between(hourly_max_salary, 19.31, 114.04)) |>
  filter(between(follower_count, 328.15, 3326613.00)) |>
  filter(views>5)
```

Introduction and data

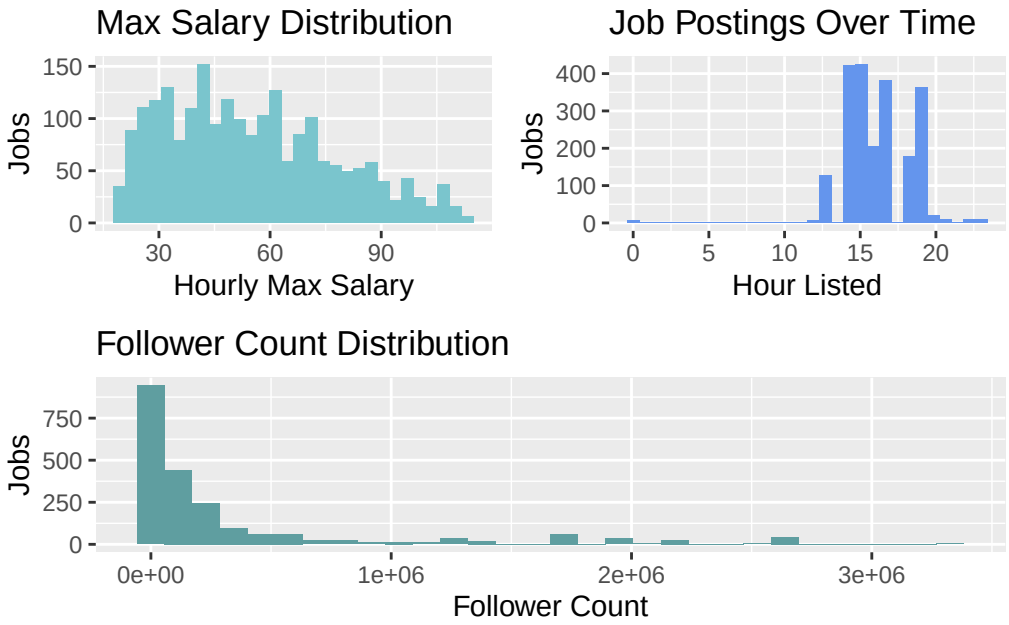
LinkedIn is a popular platform that connects companies and professionals spanning various levels of experience, and there are thousands of active job postings available on LinkedIn. Moreover, online job search services and platforms are now considered equally important for people to access a wide variety of opportunities compared to in-person job postings. With the sheer amount of postings, applicants may be overwhelmed by the vast amount of postings, and are less likely to come across some postings over others. Our primary research question is - what variables about job postings increase popularity among applicants?

This data set was created by Arsh Koneru-Ansari in July 2023, who used Python to scrape data directly from linkedin.com. The scraper code is published in their GitHub.

The data dictionary for the variable definitions can be found in the ReadMe for the data. The variables we will focus on are:

- **applies:** number of applications that have been submitted
- **views:** number of times the job posting has been viewed
- **max_salary:** maximum salary offered in the position
- **remote_allowed:** whether job permits remote work (1 = yes)
- **follower_count:** number of company followers on LinkedIn

- **`listed_time`:** time when the job was listed, in UNIX time
- **`formatted_experience_level`:** job experience level (entry level, associate, mid-senior level, director, executive, internship)
- **`type`**: type of benefit provided (Medical insurance, Dental insurance, 401(k), Paid maternity leave, Disability insurance, Vision insurance, Tuition assistance, Pension plan, Child care support, Commuter benefits, Student loan assistance)




```
sal_quantile<-quantile(linkedin_subset$hourly_max_salary, probs = c(0.05, 0.5, 0.95), na.r
follower_quantile<-quantile(linkedin_subset$follower_count, probs = c(0.05, 0.5, 0.95),na.
views_quantile<-quantile(linkedin_subset$views, probs = c(0.1,0.5, 0.95), na.rm = TRUE)
```

```
linkedin_subset_small <- linkedin_subset[, c(3,8,10)]
summary_stats <- summary(linkedin_subset_small)
print(summary_stats)
```

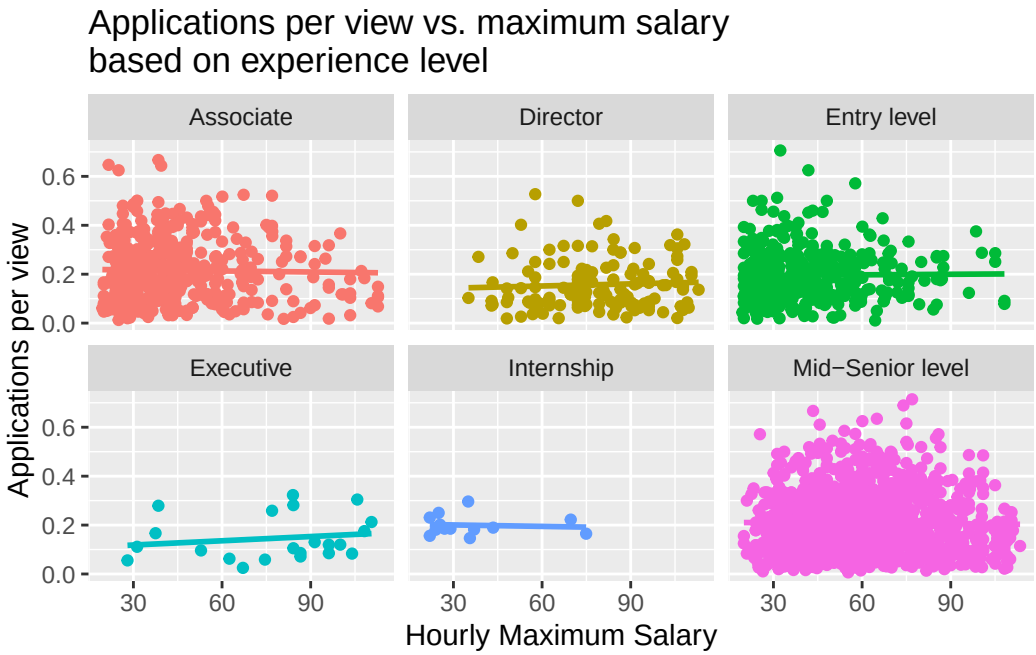
```
time_posted        follower_count     hourly_max_salary
Length:2170        Min.   :    333   Min.   : 19.50
Class :character   1st Qu.:  19296   1st Qu.: 36.06
Mode  :character   Median :  81743   Median : 52.88
                   Mean   : 344736   Mean   : 55.08
                   3rd Qu.: 286286   3rd Qu.: 70.00
```

```
Max.   :3326613   Max.   :113.46
```

Fig 1.1. The distribution of hourly maximum salary is right-skewed with jobs in the data set having a generally lower hourly maximum salary. Given the apparent skewness the center is the median hourly maximum salary of around $50 per hour. The IQR describing the spread of the 50% of the distribution is $33 per hour, demonstrating that the variability of the hourly maximum is relatively high. The histogram shows the middle 90% of the data to filter out the significant outliers so there are no apparent outliers shown in the graph.

Fig 1.2. The distribution of the hour listed is left-skewed with jobs in the data set mainly concentrated between 2:00 PM and 7:00 PM. Given the apparent skewness the center is the median hour listed at around 4:00 PM. The IQR describing the spread of the 50% of the distribution is 4 hours, showing how the variability of the hour listed is relatively high. There are apparent outliers in this graph when the hour listed is around 12:00 am.

Fig 1.3. The distribution of follower count is right-skewed with jobs in the data set having a generally lower number of followers.Given the apparent skewness the center is the median of followers of around 70 thousand followers. The IQR describing the spread of the 50% of the distribution is over 280,000 followers, which means the variability of follower count in the data set is relatively high.

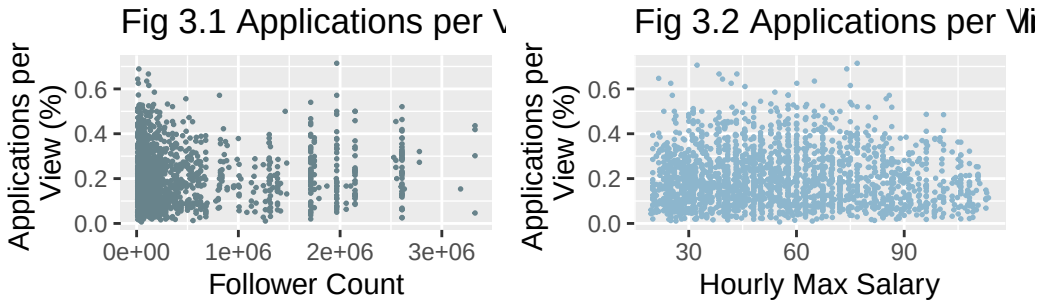


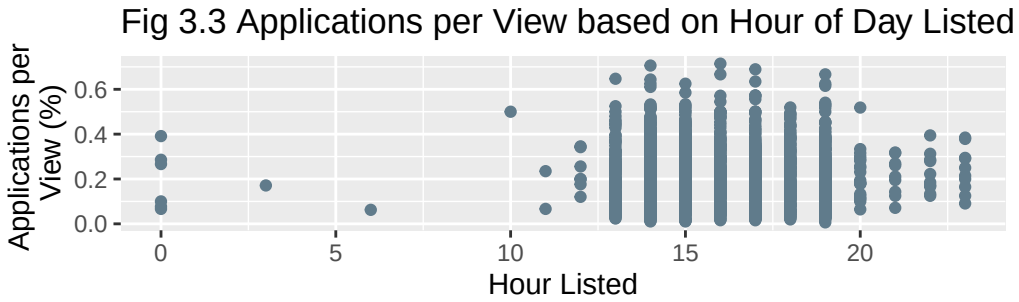

```
[1] -0.03254733
```

```r
tapply(linkedin_subset$per_applies, linkedin_subset$formatted_experience_level, summary)
```

```
$Associate
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01282 0.11697 0.19017 0.21537 0.30047 0.66667

$Director
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01887 0.08069 0.13636 0.15725 0.20840 0.52717

$`Entry level`
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.01075 0.11470 0.17752 0.19601 0.25266 0.70588

$Executive
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.02500 0.08374 0.11534 0.14591 0.20313 0.32258

$Internship
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1467  0.1810  0.1852  0.1993  0.2222  0.2963

$`Mid-Senior level`
    Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
0.006579 0.111111 0.184624 0.207172 0.282971 0.714286
```

Fig 2. There is no apparent direction or shape between the hourly maximum salary and applications per view for each of the experience levels based on the graphs. The correlation between these two variables is around -0.033, indicating the relationship is moderately weak. The distribution of applications per view based on experience level and maximum salary is mostly concentrated when the applications per view is less than 0.50 and the hourly maximum salary is less than 200 for each experience level. The mid-senior level has apparent outliers when the hourly maximum salary is greater than 200 and the NA level has outliers when the applications per view is above 0.75. There is also not as much data for some of the experience levels, including internship and executive.





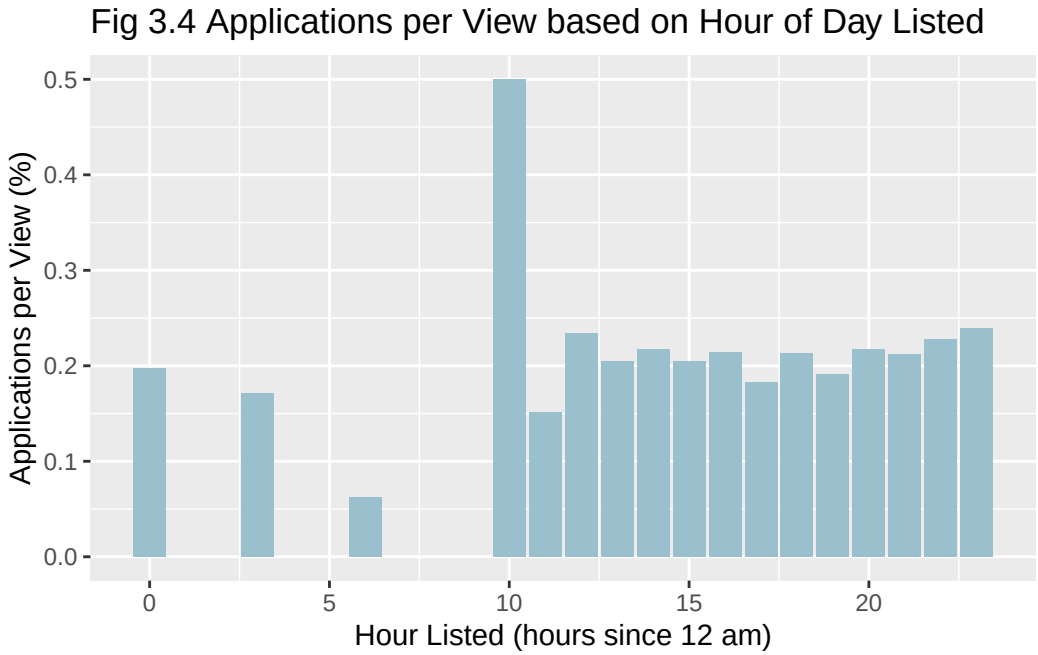


Fig 3.1. There is a slight linear correlation between follower count of a company and percentage of viewers who apply to the job. Most of the observations are concentrated to have less than 1,000,000 followers, so it would be beneficial to remove outliers that have more followers than that.

Fig 3.2. There is a slight positive linear correlation between a job's adjusted hourly maximum salary and percentage of viewers who apply to the job. Most of the observations are concentrated to have less than $200,000 for adjusted hourly max salary, so it would be beneficial to filter out outliers that are above this threshold. Additionally, it would be beneficial to remove the outlier where 100% of viewers applied to the job (per_applies = 1.0)

Fig 3.3. Convert to categorical variable, make a histogram that's colored by morning, afternoon, evening

Fig 3.4. This figure could be statistically misleading since the intervals are not consistent, and mean_per_applies is not a helpful response variable for the model. We should convert hour_listed to a categorical variable and create a pie chart instead to show the frequency of each level.

Methodology

Intro
The data provides information about job listing details such as views, applications, time posted, etc., as well as whether or not certain benefits were offered in the job listing. This information was scraped from LinkedIn on July 23 and 24, 2023.

Joining Datasets
While most of our data is from the 'job_postings' data set, we also wanted to include employee and follower count from the 'employee' data set, and the type and number of benefits listed from the 'benefits' data set. We joined all data sets together by 'company_id', and saved the data set as 'linkedin'.

Benefits
Then, we added the number of benefits to create a new "tot_benefits" predictor, which tells us the total number of benefits listed. If "remote_allowed" and "tot_benefits" had NA entries, we assumed the job did not allow remote work and did not list any benefits. Consequently, we imputed them to 0's. However, the majority of entries in 'benefits' are NA, which means no benefits were scraped from the listing. To make data from 'benefits' a useful predictor, we created a new categorical variable''if_benefits', which tells us if any or no benefits are listed. If any benefits (i.e. paid maternity leave or 401k plan) were listed, then the post is considered 'listed', and otherwise considered 'none'.

Salary
We also made assumptions about other variables to normalize predictor variables. To compare salaries even if they were listed in different formats (such as hourly pay, monthly, or yearly salary), we normalized the variable using the categorical variable 'pay_period', which tells us if the job pays its worker the 'max_salary' or 'min_salary' amount, with hourly, monthly, and annual payments. We then calculated the hourly wage given the maximum pay for hourly, monthly, and yearly pay periods. We assumed 160 hours for the monthly payments (40 hour

work week for 4 weeks), and 2080 hours for the annual payments (40 hour work week for 52 weeks). Then, we saved the new data in a variable called 'hourly_max_salary'.

Time Posted
Since the existing posted time is in different time zones, we converted the time format to EST time and only kept the hour. Then, we converted posted time from a quantitative variable to a categorical variable. Since it wouldn't make sense to have 24 different levels, we designated 4 levels: night (0 am - 5 am), morning (6 am - 11 am), afternoon (12 pm - 5 pm), and evening (6 pm - 11 pm).

Drop NA
We dropped all 'NA' values for all predictors we wanted to observe so we can keep our dataset consistent when testing different models. Specifically, we dropped NAs for 'hourly_max_salary', 'per_applies', 'follower_count', 'formatted_experience_level', 'original_listed_time', and 'remote_allowed'.

Filtering
To remove significant outliers that may affect the model's precision, we filtered and only kept the middle 90% of the data for the "hourly_max_salary" and "follower_count" variables. To generalize our results to job listings with a reasonable number of views, we filtered out job listings with less than 5 views.

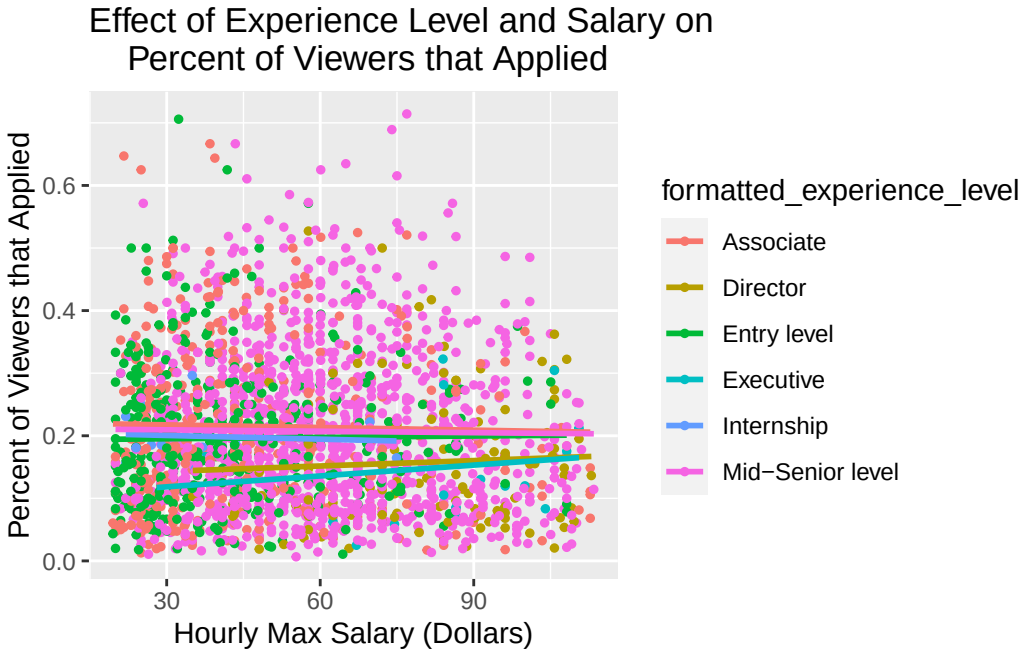
Normalize Response Variable
Because each job has been listed for varying time durations, and its view count is directly related to its application count, we decided to normalize the application count with the view count. To do so, we created a new variable 'per_applies', which divides 'applications' by 'views'. We now use 'per_applies' as our response variable.

Model Type
Because 'per_applies' is a numerical variable, a linear regression model would be most appropriate to predict the number of applications per view. As we addressed in the introduction, a person takes into consider many factors when applying to a job, so our model takes into consideration multiple predictors, including the hour of day posted, company follower count, experience level, maximum salary, ability to work remote, and if benefits are listed.

Checking interaction effect:

```
linkedin_subset |>
  ggplot(aes(x = hourly_max_salary, y = per_applies, color = formatted_experience_level))
  geom_point(size = 1) +
  geom_smooth(method = "lm", formula = y~x, se = F) +
  labs(
    x = "Hourly Max Salary (Dollars)",
    y = "Percent of Viewers that Applied",
```

```
    title = "Effect of Experience Level and Salary on
    Percent of Viewers that Applied"
  )
```



To check for interaction effects, we look at the relationship between two variables when a variable is grouped to see if the perfect of viewers that applied differs based on the experience level of the job posted. Since the lines of best fits are not parallel, there is some indication of an interaction effect, where the experience level potentially has an effect on how salary affects percent of viewers that applied.

We split the 'linkedin' dataset into training and testing data, with 75% of the data in training and 25% in testing. We then used cross-fold validation with 12 folds on the training data set to find the mean summary statistics (AIC, BIC, Adjusted R-Squared) for each model and compared the different values to find the best possible model. This process was repeated for models containing each combination of predictor variables. We set a seed of (2) when splitting and folding the data to ensure reproducibility.

```
set.seed(2)
linkedin_recipe3 <- recipe(per_applies ~ job_id + formatted_experience_level +
                             follower_count +
                            remote_allowed + hourly_max_salary,
                           data = linkedin_train) |>
```

```
    step_interact(terms = ~ hourly_max_salary:remote_allowed) |>
    update_role(job_id, new_role = "ID") |>
    step_mutate(follower_count = follower_count/1000000) |>
    step_center(hourly_max_salary, follower_count) |>
    step_dummy(all_nominal_predictors()) |>

    step_zv(all_predictors())
```

```
  set.seed(2)
  calc_model_stats <- function(x) {
    glance(extract_fit_parsnip(x)) |>
      select(adj.r.squared, AIC, BIC)
  }
```

```
  map_df(linkedin_fit_rs_full$.extracts, ~ .x[[1]][[1]]) |>
    summarise(mean_adj_rsq = mean(adj.r.squared),
              mean_aic = mean(AIC),
              mean_bic = mean(BIC))
```

```
# A tibble: 1 x 3
  mean_adj_rsq mean_aic mean_bic
         <dbl>    <dbl>    <dbl>
1       0.0233   -2096.   -2022.
```

```
  map_df(linkedin_fit_rs_red$.extracts, ~ .x[[1]][[1]]) |>
    summarise(mean_adj_rsq = mean(adj.r.squared),
              mean_aic = mean(AIC),
              mean_bic = mean(BIC))
```

```
# A tibble: 1 x 3
  mean_adj_rsq mean_aic mean_bic
         <dbl>    <dbl>    <dbl>
1       0.0237   -2101.   -2054.
```

```
  map_df(linkedin_fit_rs_red_inter$.extracts, ~ .x[[1]][[1]]) |>
    summarise(mean_adj_rsq = mean(adj.r.squared),
              mean_aic = mean(AIC),
              mean_bic = mean(BIC))
```

```
# A tibble: 1 x 3
  mean_adj_rsq mean_aic mean_bic
         <dbl>    <dbl>    <dbl>
1       0.0238   -2100.   -2041.
```

```
linkedin_fit_full <- linkedin_wflow1 |>
  fit(data = linkedin_train)

tidy(linkedin_fit_full) |>
  kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.210	0.008	27.663	0.000
hourly_max_salary	0.000	0.000	-1.335	0.182
follower_count	0.011	0.005	2.250	0.025
remote_allowed_yes	0.035	0.007	4.623	0.000
formatted_experience_level_Director	-0.047	0.015	-3.213	0.001
formatted_experience_level_Entry.level	-0.016	0.010	-1.605	0.109
formatted_experience_level_Executive	-0.060	0.029	-2.110	0.035
formatted_experience_level_Internship	-0.004	0.040	-0.089	0.929
formatted_experience_level_Mid.Senior.level	-0.006	0.008	-0.702	0.483
if_benefits_listed	-0.007	0.006	-1.088	0.277
time_posted_evening	-0.001	0.007	-0.215	0.830
time_posted_morning	-0.059	0.084	-0.693	0.488
time_posted_night	-0.054	0.060	-0.899	0.369

The adjusted R^2 value for a model including all predictors, `hourly_max_salary`, `follower_count`, `remote_allowed`, `formatted_experience_level`, `hour_listed`, and `if_benefits`, is 0.0233, and the AIC and BIC was -2095.867 and -2021.562, respectively.

Using a significance level of $\alpha = 0.10$, `follower_count`, `remote_allowed`, if the job requires director level of experience, and if the job requires executive level of experience are the only statistically significant variables, with p-values of 0.025, about 0, 0.001, and 0.035 respectively.

Then, we made a reduced model with these statistically significant variables including `follower_count`, `remote_allowed`, and `formatted_experience_level` to compare models. The adjusted R^2, AIC, and BIC were 0.0237, -2101.431, and -2053.664, respectively. Since this reduced model with less predictors has a higher adjusted R^2 and lower AIC and BIC, which all indicate a stronger or better model, we concluded that the reduced model with `follower_count`, `remote_allowed`, and `formatted_experience_level`as predictors works the better than the full model.

Lastly, we included a model with the variables from the reduced model, plus `hourly_max_salary` to explore the potential interaction effect between `hourly_max_salary` and `remote_allowed`, which we identified earlier has a potential interaction term. The adjusted R^2, AIC, and BIC are 0.0238, -2099.576, and -2041.193. Although the adjusted R-squared is higher for the reduced model with the interaction effect, the AIC and BIC were higher than those of the reduced model without the interaction effect, indicating that the reduced model without the interaction term is a stronger model. Since the adjusted R-squared for both models aren't significantly different, we decided that the reduced model without the interaction term would work better because of the lower AIC and BIC.

Overall, by comparing the adjusted R^2, AIC, and BIC values of each of the models (with all predictors, half as many predictors, and half has many predictors with an interaction term), we concluded that the reduced model with `follower_count`, `remote_allowed`, and `formatted_experience_level`as predictors works as the best model.

Results

```
linkedin_red_fit <- linkedin_wflow2 |>
  fit(data = linkedin_train)

tidy(linkedin_red_fit) |>
  kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.209	0.007	30.470	0.000
follower_count	0.010	0.005	2.164	0.031
remote_allowed_yes	0.033	0.007	4.482	0.000
formatted_experience_level_Director	-0.054	0.014	-3.939	0.000
formatted_experience_level_Entry.level	-0.016	0.010	-1.637	0.102
formatted_experience_level_Executive	-0.068	0.028	-2.402	0.016
formatted_experience_level_Internship	-0.003	0.040	-0.064	0.949
formatted_experience_level_Mid.Senior.level	-0.009	0.008	-1.138	0.255

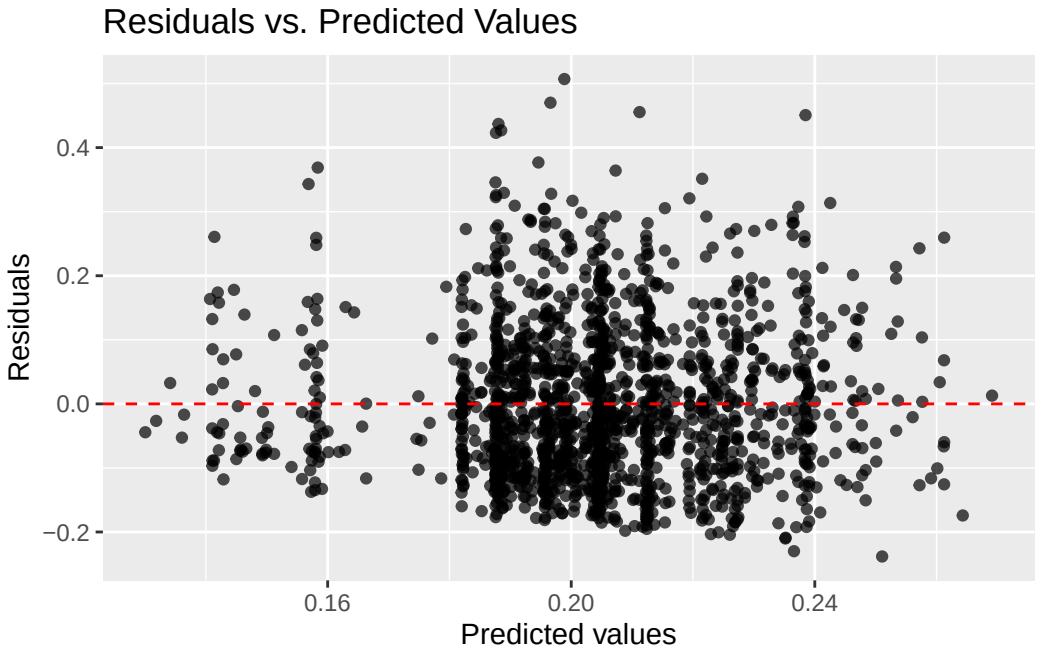
$per\_applies = 0.209 + 0.010(follower\_count(millions)) + 0.033(remote\_allowed) - 0.054(Director) - 0.016(Entry\_level) - 0.068(Executive) - 0.003(Internship) - 0.009(Mid\_Senior\_level)$

Linearity:

```
linkedin_fit_red <- linear_reg() |>
  set_engine("lm") |>
```

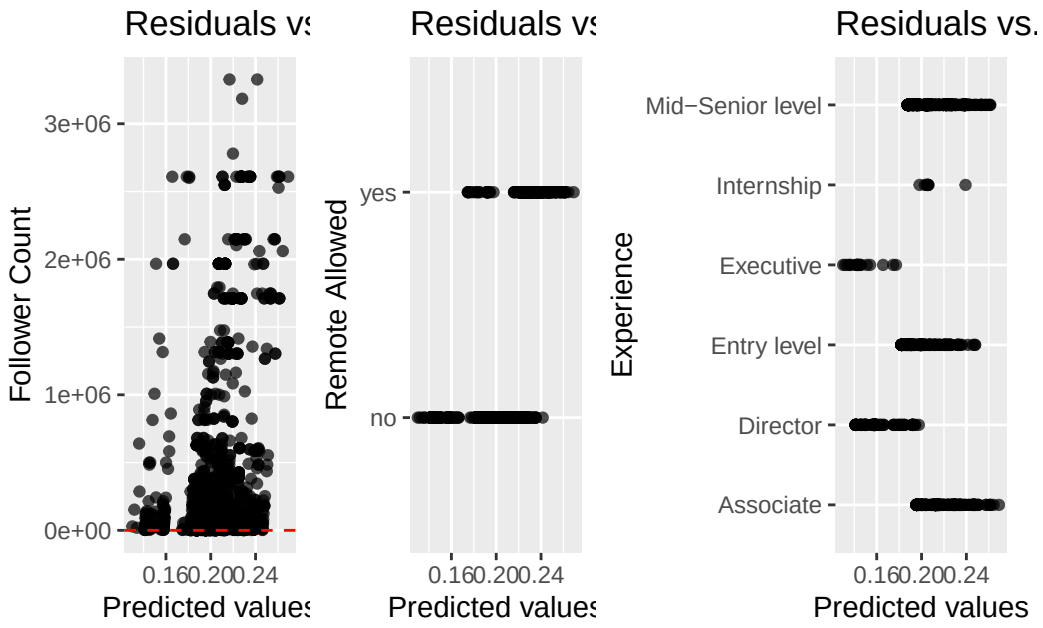
```
  fit(per_applies ~ job_id + remote_allowed +
                            formatted_experience_level + follower_count, data = linkedin_
```

```
linkedin_red_aug <- augment(linkedin_fit_red$fit)
ggplot(data = linkedin_red_aug, aes(x = .fitted, y = .resid)) +
geom_point(alpha = 0.7) +
geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
labs(x = "Predicted values",
     y = "Residuals",
     title = "Residuals vs. Predicted Values")
```




```
r2 <- ggplot(data = linkedin_red_aug, aes(x = .fitted, y = follower_count)) +
geom_point(alpha = 0.7) +
geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
labs(x = "Predicted values",
     y = "Follower Count",
     title = "Residuals vs. Follower Count")
r3 <- ggplot(data = linkedin_red_aug, aes(x = .fitted, y = remote_allowed)) +
geom_point(alpha = 0.7) +
geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
```

```
labs(x = "Predicted values",
     y = "Remote Allowed",
     title = "Residuals vs. Remote")
r4 <- ggplot(data = linkedin_red_aug, aes(x = .fitted, y = formatted_experience_level)) +
geom_point(alpha = 0.7) +
geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
labs(x = "Predicted values",
     y = "Experience",
     title = "Residuals vs. Experiences")
r2+r3+r4
```



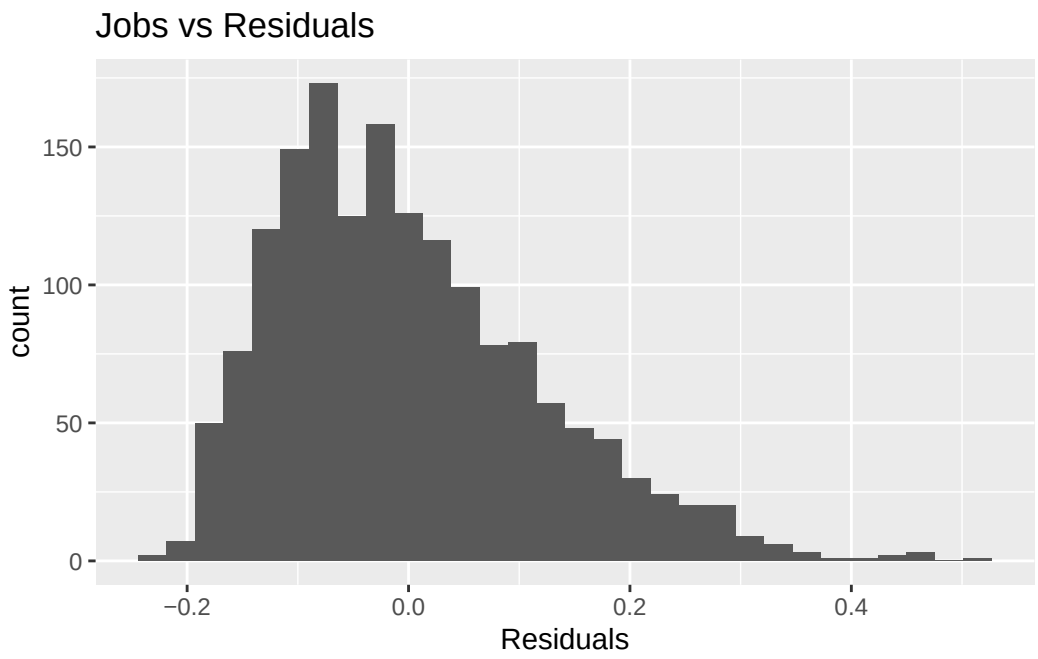
Since the plot of the residuals vs. predicted values do not have a discernible pattern and the plots of the residuals vs. each predictor do not have a discernible pattern, linearity is met.

Constant Variance:

The vertical spread of the residuals is not constant in the plot of the residuals vs. predicted so the constant variance condition is not satisfied.

Normality:

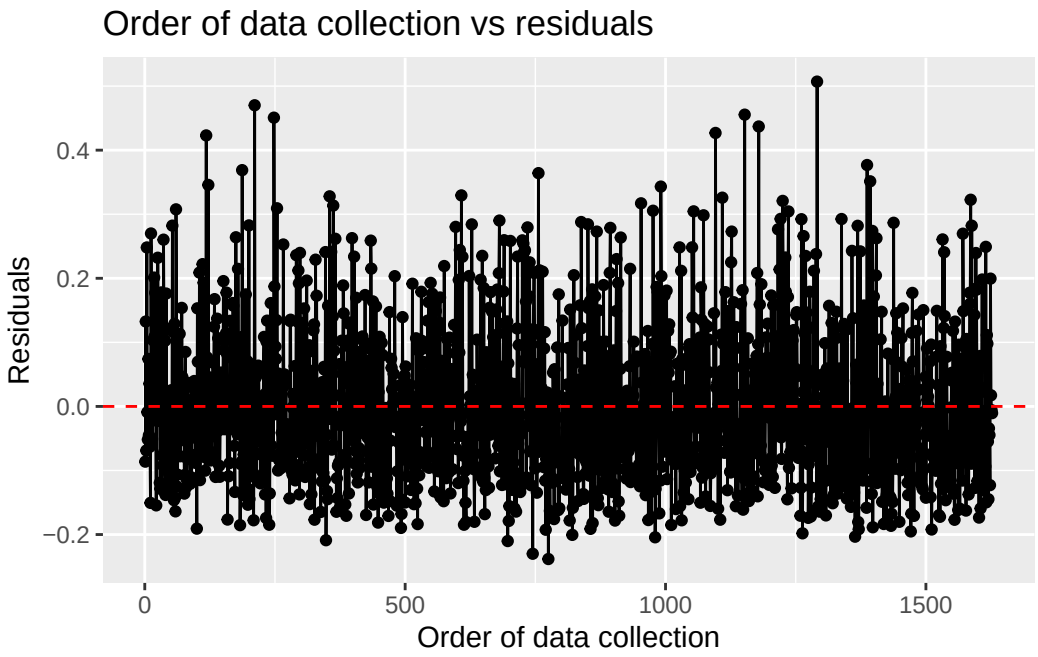
```
ggplot(data = linkedin_red_aug, aes(x = .resid)) +
  geom_histogram() +
  labs(title = "Jobs vs Residuals",
       x = "Residuals")
```



The distribution of the residuals is unimodal but not symmetric, so the normality condition is not satisfied. However, the sample size is large enough to relax this condition since it is not satisfied.

Independence:

```
ggplot(linkedin_red_aug, aes(y = .resid, x = 1:nrow(linkedin_red_aug))) +
  geom_line() +
  geom_point() +
  geom_hline(yintercept = 0, color = "red", linetype = "dashed") +
  labs(x = "Order of data collection",
       y = "Residuals",
       title = "Order of data collection vs residuals")
```



Independence is not satisfied because we are looking at company-related attributes including the company follower count and the postings for different jobs from the same companies may be dependent of each other. Additionally, the dataset spans two days that are months apart so the observations within the same time period are not independent since the new job postings may be dependent from the older ones. However, there is clear pattern in the residuals vs. order of data collection plot.

```
vif(linkedin_fit_red$fit)
```

```
                                  job_id
                                1.041149
                          remote_allowedyes
                                1.013814
         formatted_experience_levelDirector
                                1.246982
      formatted_experience_levelEntry level
                                1.620469
        formatted_experience_levelExecutive
                                1.049967
       formatted_experience_levelInternship
                                1.026203
formatted_experience_levelMid-Senior level
```

```
      1.767394
follower_count
      1.023741
```

Multicollinearity occurs where there are very high correlations among two or more predictor variables, and we need to check for multicollinearity because it causes a loss in precision in our estimates of the regression coefficients. It is a concern when VIF is greater than 10 for a predictor, which is not the case in our model.

```
# training data
linkedin_train_pred <- predict(linkedin_red_fit, linkedin_train) |>
  bind_cols(linkedin_train)

rmse_train <- rmse(linkedin_train_pred, truth = per_applies, estimate = .pred)

# testing data
linkedin_test_pred <- predict(linkedin_red_fit, linkedin_test) |>
  bind_cols(linkedin_test)

rmse_test <- rmse(linkedin_test_pred, truth = per_applies, estimate = .pred)

bind_rows(rmse_train, rmse_test) |>
  kable(digits = 4)
```

.metric	.estimator	.estimate
rmse	standard	0.1189
rmse	standard	0.1193

The RMSE of the training data is 0.1189, and the RMSE of the testing data is 0.1193. Significantly lower RMSEs for training data compared to the testing data could be a sign of model overfit, which means that the model fits the training data too well where it doesn't model new unseen data well. However, since the RMSE values for the training and testing data are very close, this shows no evidence of model overfit.

```
glance(linkedin_red_fit)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic     p.value    df logLik    AIC    BIC
      <dbl>         <dbl> <dbl>     <dbl>       <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1    0.0280        0.0238 0.119      6.67     8.40e-8     7  1156. -2293. -2245.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Looking at the how the follower count, whether the job allows remote, and the listed job experience level affected the ratio of applications to views, there is no significant relationship between these variables, since the adjusted R^2 s 0.0280, which is very low. This means that less than 3% of the variation in the percent of viewers that applied is a result of the variation in the predictor variables we are studying, indicating no relationship.We tried different combinations of variables to see if there was any relationship between the variables with percent of viewers that applied.

Discussion + Conclusion

While we had expected the percent of viewers that applied for each job to increase as the maximum salary and number of followers increase, our model and EDA have demonstrated how there is no relationship between these variables, along with having a remote option, the job experience level, the hour at which it was posted, and if the benefits were posted, and our response variable of the percent of viewers who applied.

Some of the limitations with our analysis may include assumption errors with the number of hours the job would work for the jobs listed as yearly and monthly, the filtering of outliers and deciding the thresholds for outliers, model complexity where we are fitting many predictor variables, and the lack of data considering how long the jobs were listed.

Some ways our analysis could be improved would be filtering out the outliers better, deciding the thresholds for views, follower count, salaries, etc, that would avoid skewing the data.

Potential issues relating to the data would include the time scraped from LinkedIn, since many of the original listed times and the listed times (the time it was scraped) is exactly the same, which is not meaningful in determining the number of applications over a certain amount of time.