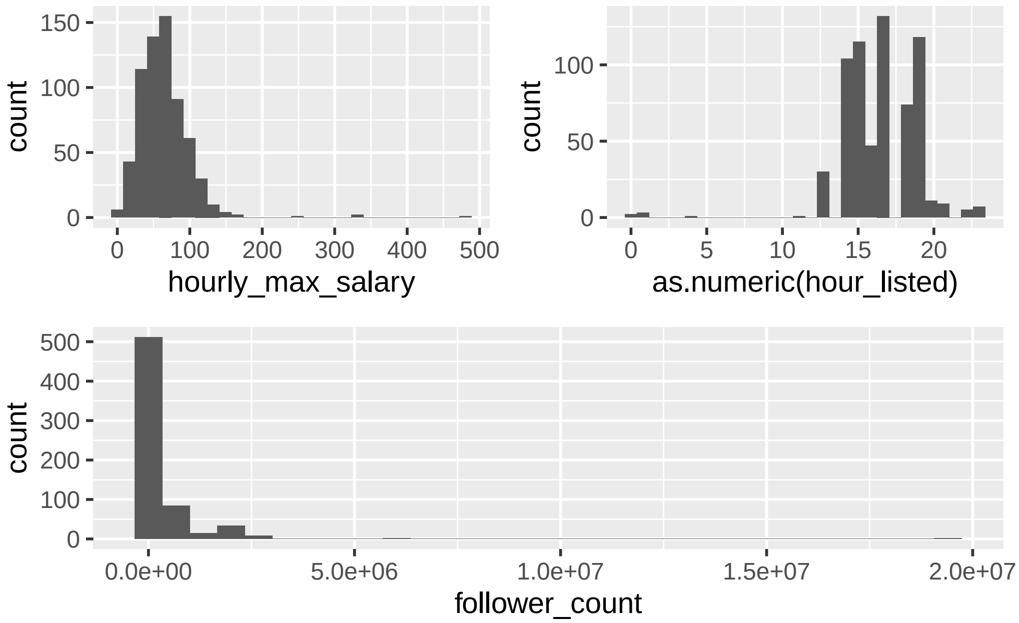
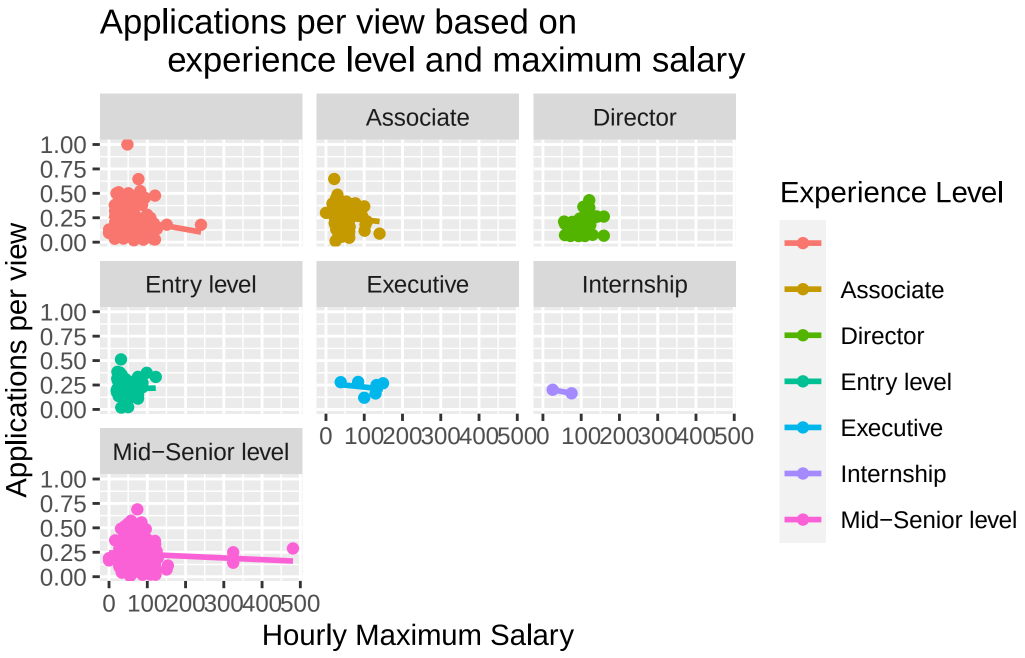
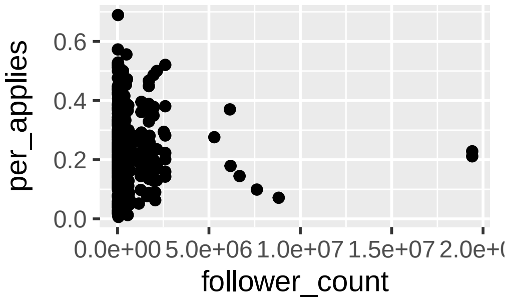
Factors Impacting Number of LinkedIn Job Applications per Post View

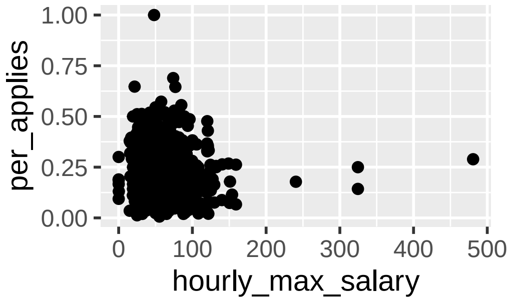
JRLK - Jessie Ringness, Rebekah Kim, Laura Cai, Karen Dong

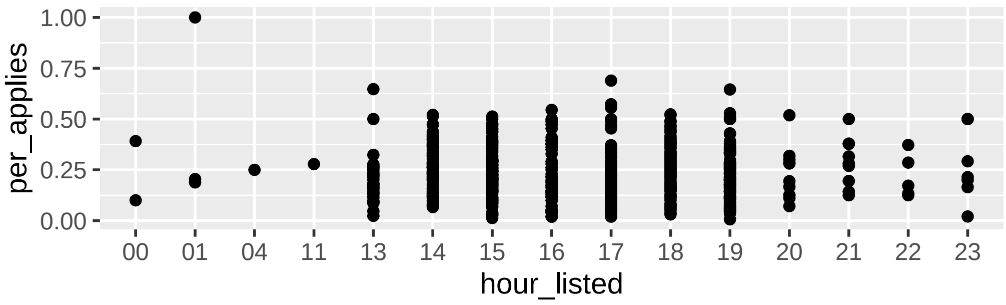
2023-11-14

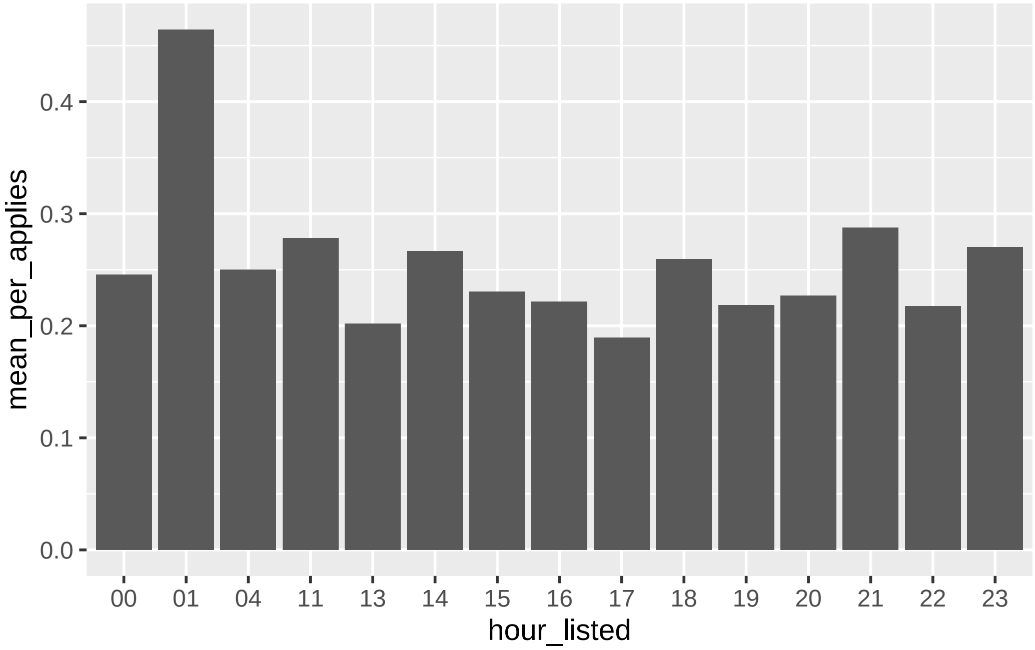



Applications per view based on
experience level and maximum salary
Associate
Director
Entry level
Executive
Internship
Mid−Senior level
Applications per view
1.00
0.75
0.50
0.25
0.00
0 100200300400500
Hourly Maximum Salary
Experience Level
Associate
Director
Entry level
Executive
Internship
Mid−Senior level


per_applies
0.6
0.4
0.2
0.0
0.0e+00
5.0e+06
1.0e+07
1.5e+07
2.0e+0(
follower_count


per_applies
1.00
0.75
0.50
0.25
0.00
0
100
200
300
400
500
hourly_max_salary


per_applies
1.00
0.75
0.50
0.25
0.00
00
01
04
11
13
14
15
16
17
18
19
20
21
22
23
hour_listed



Introduction

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 (https://github.com/ArshKA/LinkedIn-Job-Scraper#jobs).

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)

Another potential interaction effect is the number of views and experience level. Different positions are more sought after on LinkedIn over others, depending on viewers' backgrounds. The number of applications increases per number of views for a position requiring associate experience increases at a more rapid rate than those for other positions, although all positions have some sort of impact associated with views and applications.

Methodology

While the bulk of our data was found in the 'job_postings' data set, we also wanted to include employee count and follower count data in the 'employee' data set and the type and number of benefits listed on the post found in the 'benefits' data set. We needed to manipulate the data in 'benefits' to create a useful predictor as the majority of the data was NA, meaning no benefits could be found from the scraped data. To make the data from 'benefits' a useful predictor, we created a new categorical variable called ''if_benefits' where if a benefit (such as paid maternity leave or a 401k plan) was listed on the post, then the post was considered as having benefits 'listed', and otherwise was listed as 'none' listed. We joined all data sets together by 'company_id', and saved the data set as 'linkedin'.

We also made some assumptions about other variables to normalize predictor variables. The categorical variable 'pay_period' contained data on when the job would pay its worker the 'max_salary' or 'min_salary' amount, with hourly, monthly, and annual payments as the different levels. To normalize the 'max_salary' amount, we 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). We saved the new data in a variable called 'hourly_max_salary'.

We then dropped all 'NA' values for all predictors we wanted to observe so that we could keep our dataset consistent when testing different models, meaning NA values for 'hourly_max_salary', 'per_applies', 'follower_count', 'formatted_experience_level', 'original_listed_time', and 'remote_allowed'.

Because the number of views an application gets is directly related to the number of applications and the jobs have been listed for varying durations of time, we decided to normalize the number of applications with the views. To do so, we created a new variable ‘per_applies‘. We then used ‘per_applies‘ as our response variable.

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 different factors when applying to a job, so our model takes into consideration multiple predictors, including the hour the job was posted, the number of followers the company has, the job experience level, the maximum salary, ability to work remote, and if benefits are listed.

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.

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

```
# A tibble: 2 x 6
  .metric .estimator   mean     n std_err .config
  <chr>   <chr>       <dbl> <int>   <dbl> <chr>
1 rmse    standard   0.127     12 0.00531 Preprocessor1_Model1
2 rsq     standard   0.0356    12 0.0138  Preprocessor1_Model1
```

```
# A tibble: 1 x 3
  mean_adj_rsq mean_aic mean_bic
         <dbl>    <dbl>    <dbl>
1      0.00761    -574.    -477.
```

The adjusted R^2 value for a model including `hourly_max_salary`, `follower_count`, `remote_allowed`, `formatted_experience_level`, `hour_listed`, and `if_benefits` was 0.0076. We removed variables one by one, but each model resulted in a lower R^2 value, indicating that a model including all mentioned variables is the best.

```
linkedin_fit <- linkedin_wflow1 |>
  fit(data = linkedin_test)
linkedin_test_pred <- predict(linkedin_fit, linkedin_test) |>
  bind_cols(linkedin_test)
```

```
rsq(linkedin_test_pred, truth = per_applies, estimate = .pred)
```

```
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rsq     standard       0.254
```

Results

```
linkedin_yr_rec <- recipe(per_applies ~ job_id + hourly_max_salary + follower_count + remo
  update_role(job_id, new_role = "ID") |>
  step_naomit(all_predictors()) |>
  step_dummy(all_nominal_predictors()) |>
  step_zv(all_predictors())
```

```
#specify the model
linkedin_yr_spec <- linear_reg() |>
  set_engine("lm")

#build model workflow
linkedin_yr_workflow <- workflow() |>
  add_model(linkedin_yr_spec) |>
  add_recipe(linkedin_yr_rec)

# fit the model
linkedin_yr_fit <- linkedin_yr_workflow |>
  fit(data = linkedin_subset)

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

term	estimate	std.error	statistic	p.value
(Intercept)	0.260	0.095	2.730	0.007
hourly_max_salary	0.000	0.000	-1.462	0.145
follower_count	0.000	0.000	-0.396	0.692
formatted_experience_level_Associate	0.004	0.027	0.158	0.875
formatted_experience_level_Director	-0.030	0.038	-0.786	0.433
formatted_experience_level_Entry.level	-0.002	0.027	-0.092	0.927
formatted_experience_level_Executive	0.107	0.132	0.806	0.421

term	estimate	std.error	statistic	p.value
formatted_experience_level_Internship	0.003	0.138	0.020	0.984
formatted_experience_level_Mid.Senior.level	0.027	0.019	1.453	0.147
hour_listed_X01	0.221	0.120	1.851	0.065
hour_listed_X04	-0.069	0.205	-0.335	0.738
hour_listed_X13	-0.052	0.103	-0.510	0.610
hour_listed_X14	0.028	0.094	0.302	0.763
hour_listed_X15	-0.006	0.094	-0.067	0.946
hour_listed_X16	-0.037	0.096	-0.385	0.700
hour_listed_X17	-0.059	0.094	-0.625	0.533
hour_listed_X18	0.005	0.094	0.052	0.958
hour_listed_X19	-0.045	0.093	-0.484	0.629
hour_listed_X20	-0.054	0.113	-0.480	0.631
hour_listed_X21	0.093	0.113	0.820	0.413
hour_listed_X22	-0.054	0.130	-0.419	0.675
hour_listed_X23	0.094	0.109	0.857	0.392

```r
glance(linkedin_yr_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     0.122        0.0555 0.129      1.83  0.0159    21   199. -352. -267.
# i 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Looking at the how the maximum salary, follower count, whether the job allows remote, the experience level, if there were benefits listed, and the hour at which the job was listed all affected the ratio of applications to views, there is no significant relationship between these variables, since the adjusted Rˆ2 s 0.0589, which is very low. This means that about 6% 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.

Data dictionary

The data dictionary can be found here.