463 Machine Learning HW 1

Group 9: Grace Chen, Vanessa Chen, Amanda Lee, Sheryl Xu

Business Question

Yelp Dataset

Specializing Influence Marketing

Data size: 5966 restaurants

Four metropolitan areas

Q1 **Regress average star rating on the number of elite users (elite_cnt), price levels (price_level is a categorical variable with 4 levels), metropolitan area (metro, Charlotte, Phoenix, Pittsburgh, Las Vegas) and business age (biz_age) in days (M = 2237.367, SD =1384.987). Report output.**

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7119 -0.4773  0.0815  0.5679  1.8011

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.354e+00  3.332e-02 100.649  < 2e-16 ***
elite_cnt           2.393e-03  1.990e-04  12.026  < 2e-16 ***
price_levelprice_2  2.444e-01  2.052e-02  11.908  < 2e-16 ***
price_levelprice_3  2.084e-01  5.851e-02   3.561 0.000372 ***
price_levelprice_4  4.913e-01  9.273e-02   5.298 1.21e-07 ***
metroPhoenix_area   1.043e-01  3.081e-02   3.387 0.000710 ***
metroPittsburgh     1.291e-01  4.114e-02   3.137 0.001714 **
metroVegas_area     8.726e-02  3.221e-02   2.709 0.006770 **
biz_age            -8.795e-05  7.396e-06 -11.893  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7605 on 5957 degrees of freedom
Multiple R-squared:  0.07706,	Adjusted R-squared:  0.07582
F-statistic: 62.17 on 8 and 5957 DF,  p-value: < 2.2e-16
```

Q2

Test the overall significance of the model by stating the null and alternative, P - value and decision.

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7119 -0.4773  0.0815  0.5679  1.8011

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.354e+00  3.332e-02 100.649  < 2e-16 ***
elite_cnt           2.393e-03  1.990e-04  12.026  < 2e-16 ***
price_levelprice_2  2.444e-01  2.052e-02  11.908  < 2e-16 ***
price_levelprice_3  2.084e-01  5.851e-02   3.561 0.000372 ***
price_levelprice_4  4.913e-01  9.273e-02   5.298 1.21e-07 ***
metroPhoenix_area   1.043e-01  3.081e-02   3.387 0.000710 ***
metroPittsburgh     1.291e-01  4.114e-02   3.137 0.001714 **
metroVegas_area     8.726e-02  3.221e-02   2.709 0.006770 **
biz_age            -8.795e-05  7.396e-06 -11.893  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7605 on 5957 degrees of freedom
Multiple R-squared:  0.07706,   Adjusted R-squared:  0.07582
F-statistic: 62.17 on 8 and 5957 DF,  p-value: < 2.2e-16
```

H_0: All regression coefficients are equal to zero ($\beta_1 = \beta_2 = \ldots = \beta_8 = 0$

H_1: At least one regression coefficient is not equal to zero

Decision: at least on p-value is less than 0.05.

Conclusion: Since the p-value is less than 0.05, we reject the null hypothesis. There is sufficient evidence to conclude that the model is statistically significant at the 95% confidence level.

Q3 Report the estimated regression equation.

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7119 -0.4773  0.0815  0.5679  1.8011

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.354e+00  3.332e-02 100.649  < 2e-16 ***
elite_cnt           2.393e-03  1.990e-04  12.026  < 2e-16 ***
price_levelprice_2  2.444e-01  2.052e-02  11.908  < 2e-16 ***
price_levelprice_3  2.084e-01  5.851e-02   3.561 0.000372 ***
price_levelprice_4  4.913e-01  9.273e-02   5.298 1.21e-07 ***
metroPhoenix_area   1.043e-01  3.081e-02   3.387 0.000710 ***
metroPittsburgh     1.291e-01  4.114e-02   3.137 0.001714 **
metroVegas_area     8.726e-02  3.221e-02   2.709 0.006770 **
biz_age            -8.795e-05  7.396e-06 -11.893  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7605 on 5957 degrees of freedom
Multiple R-squared:  0.07706,   Adjusted R-squared:  0.07582
F-statistic: 62.17 on 8 and 5957 DF,  p-value: < 2.2e-16
```

biz.stars = 3.3540

+ 0.002393×(elite_cnt)

+ 0.2444×(price_levelPrice_2)

+ 0.2084×(price_levelPrice_3)

+ 0.4913×(price_levelPrice_4)

+ 0.1043×(metroPhoenix_area)

+ 0.1291×(metroPittsburgh)

+ 0.08726×(metroVegas_area)

- 0.00008795×(biz_age)

Q4 What fraction of variation in average star rating is explained by the terms in this model? Comment on the magnitude and the implication.

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.7119 -0.4773  0.0815  0.5679  1.8011 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)          3.354e+00  3.332e-02 100.649  < 2e-16 ***
elite_cnt            2.393e-03  1.990e-04  12.026  < 2e-16 ***
price_levelprice_2   2.444e-01  2.052e-02  11.908  < 2e-16 ***
price_levelprice_3   2.084e-01  5.851e-02   3.561 0.000372 ***
price_levelprice_4   4.913e-01  9.273e-02   5.298 1.21e-07 ***
metroPhoenix_area    1.043e-01  3.081e-02   3.387 0.000710 ***
metroPittsburgh      1.291e-01  4.114e-02   3.137 0.001714 ** 
metroVegas_area      8.726e-02  3.221e-02   2.709 0.006770 ** 
biz_age             -8.795e-05  7.396e-06 -11.893  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7605 on 5957 degrees of freedom
Multiple R-squared:  0.07706,	Adjusted R-squared:  0.07582 
F-statistic: 62.17 on 8 and 5957 DF,  p-value: < 2.2e-16
```

Multiple R-squared: 0.07706

- This means that the predictors in the model (elite_cnt, price_level, metro, and biz_age) collectively explain approximately 7.7% of the variation in the restaurants’ average star ratings.
- Magnitude: 7.7% is low, indicating that while these predictors do have a statistically significant relationship with star ratings, the majority of the variation (over 90%) remains unexplained.
- Implication: The model needs improvement. Adding more variables may capture a larger portion of variability in ratings.

Q5 Are there differences in average star rating between price levels? Use your model in part 1 to answer the question. State the null and alternative, P-value and decision using the .05 level.

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.7119 -0.4773  0.0815  0.5679  1.8011

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)
(Intercept)          3.354e+00  3.332e-02 100.649  < 2e-16 ***
elite_cnt            2.393e-03  1.990e-04  12.026  < 2e-16 ***
price_levelprice_2   2.444e-01  2.052e-02  11.908  < 2e-16 ***
price_levelprice_3   2.084e-01  5.851e-02   3.561 0.000372 ***
price_levelprice_4   4.913e-01  9.273e-02   5.298 1.21e-07 ***
metroPhoenix_area    1.043e-01  3.081e-02   3.387 0.000710 ***
metroPittsburgh      1.291e-01  4.114e-02   3.137 0.001714 **
metroVegas_area      8.726e-02  3.221e-02   2.709 0.006770 **
biz_age             -8.795e-05  7.396e-06 -11.893  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7605 on 5957 degrees of freedom
Multiple R-squared:  0.07706,    Adjusted R-squared:  0.07582
F-statistic: 62.17 on 8 and 5957 DF,  p-value: < 2.2e-16
```

$$H_0: \quad \beta_{\text{Price}_2} = 0, \quad \beta_{\text{Price}_3} = 0, \quad \beta_{\text{Price}_4} = 0$$

$$H_a: \quad \text{At least one of } \beta_{\text{Price}_2}, \beta_{\text{Price}_3}, \beta_{\text{Price}_4} \neq 0$$

Decision: **Because the p-values for all price-level coefficients are <0.05, and the overall F-test for “price” as a factor would also yield a p-value <0.05, reject H0.**

Conclusion: **Price level is positively correlated with the average star rating. since price level 1 is the dummy variable.**

Among price level 2,3 and 4, price level 4 has the highest affect on the rating, indicating consumers might hold the belief “you are paying what you get”

Q6 Does the model in part 1 provide evidence that business age affects average star rating? State the null and alternative, P-value and decision using the .05 level.

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.7119 -0.4773  0.0815  0.5679  1.8011 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)          3.354e+00  3.332e-02 100.649  < 2e-16 ***
elite_cnt            2.393e-03  1.990e-04  12.026  < 2e-16 ***
price_levelprice_2   2.444e-01  2.052e-02  11.908  < 2e-16 ***
price_levelprice_3   2.084e-01  5.851e-02   3.561 0.000372 ***
price_levelprice_4   4.913e-01  9.273e-02   5.298 1.21e-07 ***
metroPhoenix_area    1.043e-01  3.081e-02   3.387 0.000710 ***
metroPittsburgh      1.291e-01  4.114e-02   3.137 0.001714 ** 
metroVegas_area      8.726e-02  3.221e-02   2.709 0.006770 ** 
biz_age             -8.795e-05  7.396e-06 -11.893  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7605 on 5957 degrees of freedom
Multiple R-squared:  0.07706,	Adjusted R-squared:  0.07582 
F-statistic: 62.17 on 8 and 5957 DF,  p-value: < 2.2e-16
```

$H_0 : \beta_{\text{biz\_age}} = 0$

Business age has no effect on average star rating.

$H_a : \beta_{\text{biz\_age}} \neq 0$

Business age has an effect on average star rating.

Decision: Because the p-value is <0.05, reject H0.

Conclusion: The business age is negatively correlated with the average star rating. However, since the coefficient is only -0.00008795, the effect is nearly negligible.

Q7 Keep the number of elite users and the price levels as the only predictors on average star ratings in the model. Add an appropriate transformation to the model to allow for a U-shaped effect from the number of elite users on average star rating. Use at least two additional models (I suggest estimating log and quadratic models).

Baseline Model:

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level, data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-3.4702 -0.5000  0.0752  0.5591  1.7556

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.2443960  0.0144655 224.285  < 2e-16 ***
elite_cnt           0.0017778  0.0001928   9.222  < 2e-16 ***
price_levelprice_2  0.2691194  0.0206603  13.026  < 2e-16 ***
price_levelprice_3  0.2345873  0.0591306   3.967 7.36e-05 ***
price_levelprice_4  0.4995072  0.0936900   5.331 1.01e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.77 on 5961 degrees of freedom
Multiple R-squared:  0.0531,	Adjusted R-squared:  0.05247
F-statistic: 83.58 on 4 and 5961 DF,  p-value: < 2.2e-16
```

Log Model:

```
Call:
lm(formula = biz.stars ~ log(elite_cnt + 1) + price_level, data = biz)

Residuals:
     Min       1Q   Median       3Q      Max
-2.36162 -0.48773  0.04514  0.52580  1.99267

Coefficients:
                   Estimate Std. Error t value Pr(>|t|)
(Intercept)        3.007333   0.019142 157.103  < 2e-16 ***
log(elite_cnt + 1) 0.149115   0.007633  19.535  < 2e-16 ***
price_levelprice_2 0.190472   0.020679   9.211  < 2e-16 ***
price_levelprice_3 0.176814   0.057400   3.080  0.00208 **
price_levelprice_4 0.418729   0.091376   4.582 4.69e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7518 on 5961 degrees of freedom
Multiple R-squared:  0.09738,	Adjusted R-squared:  0.09677
F-statistic: 160.8 on 4 and 5961 DF,  p-value: < 2.2e-16
```

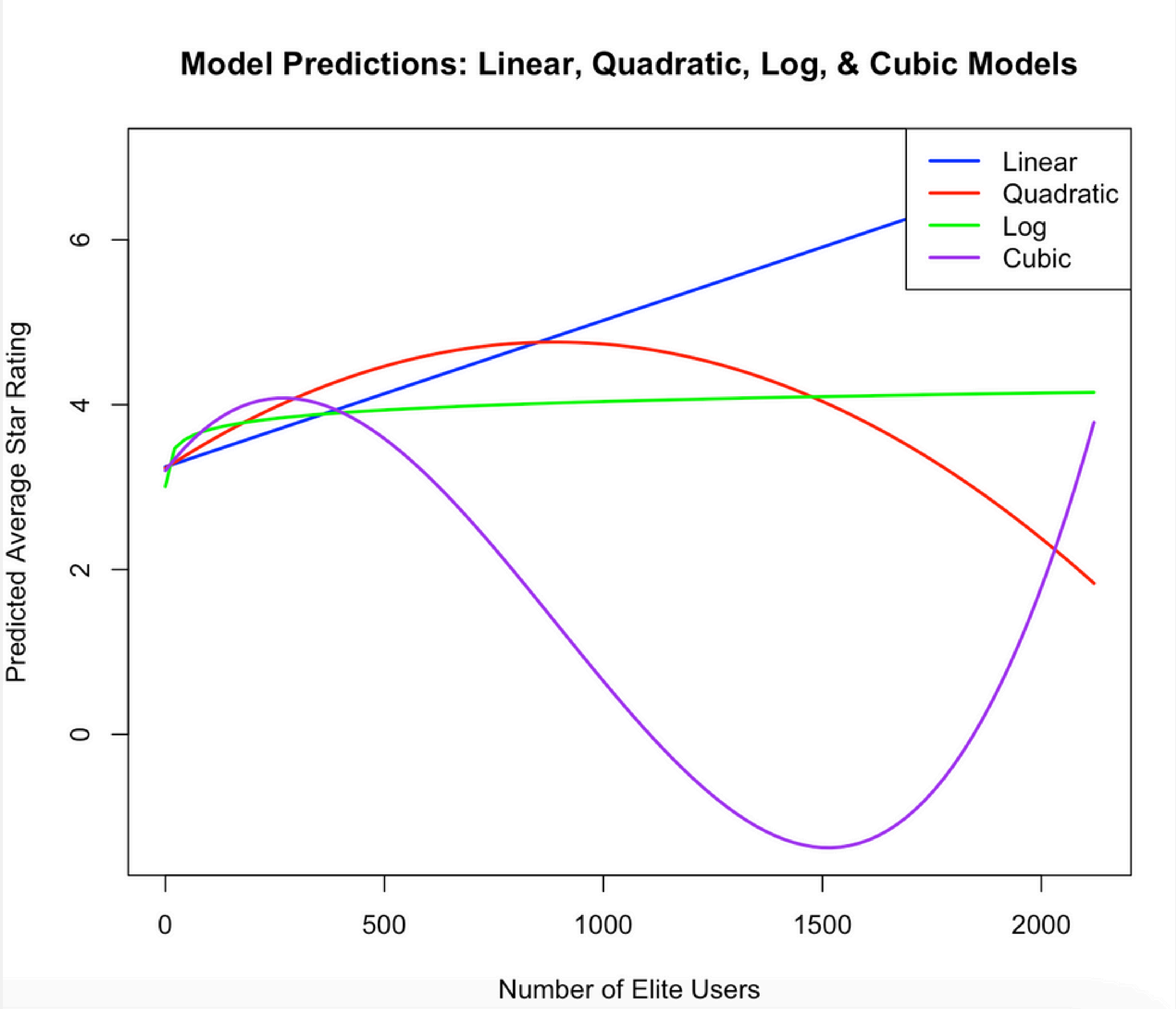
Quadratic Model:

```
Call:
lm(formula = biz.stars ~ elite_cnt + I(elite_cnt^2) + price_level,
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-2.4809 -0.4820  0.0655  0.5490  1.7715

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.229e+00  1.451e-02 222.510  < 2e-16 ***
elite_cnt           3.441e-03  2.771e-04  12.418  < 2e-16 ***
I(elite_cnt^2)     -1.934e-06  2.326e-07  -8.312  < 2e-16 ***
price_levelprice_2  2.455e-01  2.074e-02  11.837  < 2e-16 ***
price_levelprice_3  2.213e-01  5.882e-02   3.762  0.00017 ***
price_levelprice_4  4.369e-01  9.346e-02   4.674 3.02e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7657 on 5960 degrees of freedom
Multiple R-squared:  0.06395,	Adjusted R-squared:  0.06317
F-statistic: 81.44 on 5 and 5960 DF,  p-value: < 2.2e-16
```

Q7 **Keep the number of elite users and the price levels as the only predictors on average star ratings in the model. Add an appropriate transformation to the model to allow for a U-shaped effect from the number of elite users on average star rating. Use at least two additional models (I suggest estimating log and quadratic models).**

Cubic Model:

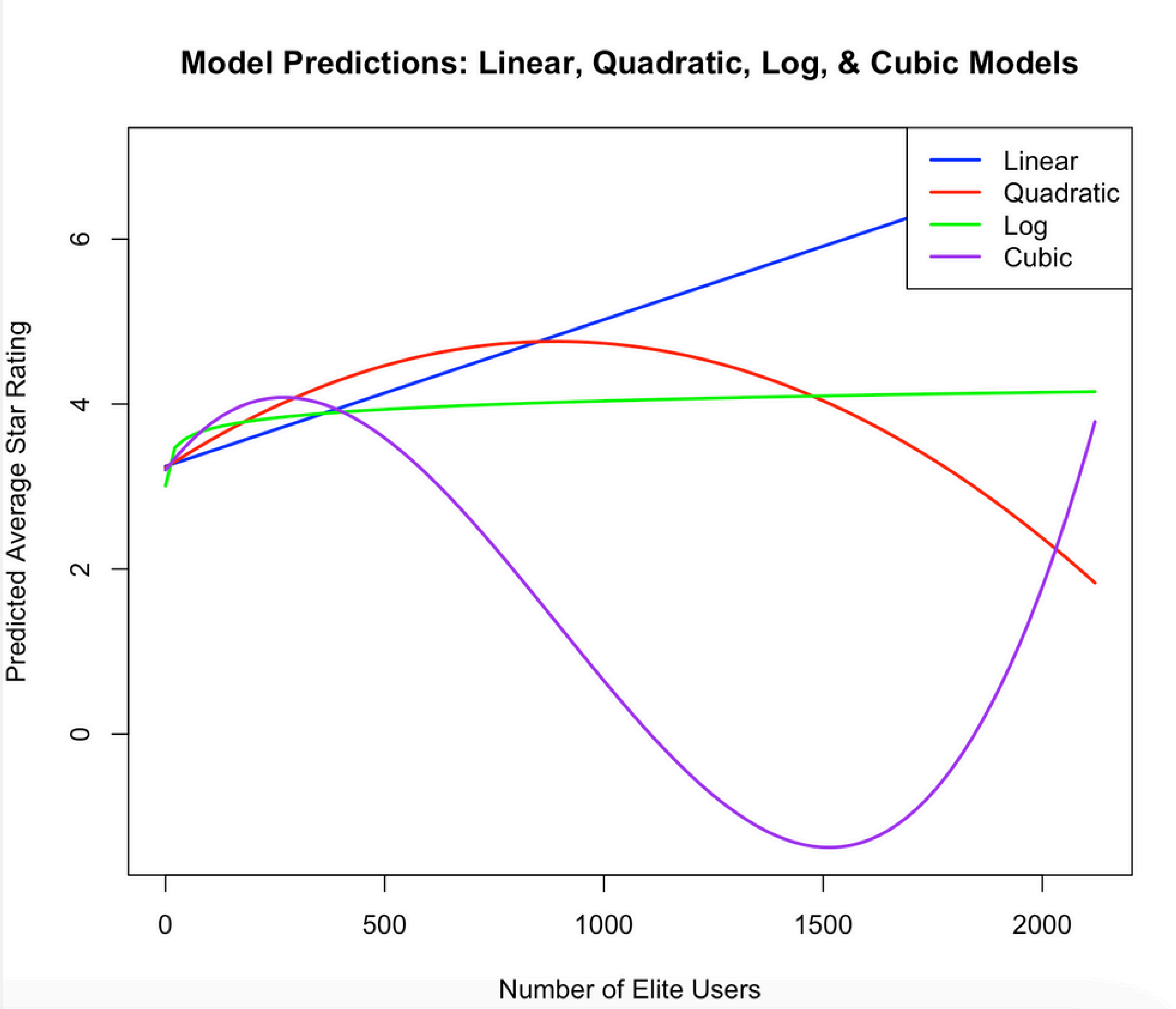
```
Call:
lm(formula = biz.stars ~ elite_cnt + I(elite_cnt^2) + I(elite_cnt^3) +
    price_level, data = biz)

Residuals:
     Min       1Q   Median       3Q      Max
-2.42984 -0.47711  0.05344  0.54236  1.96495

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.200e+00  1.465e-02 218.482  < 2e-16 ***
elite_cnt           6.943e-03  4.353e-04  15.949  < 2e-16 ***
I(elite_cnt^2)     -1.517e-05  1.297e-06 -11.694  < 2e-16 ***
I(elite_cnt^3)      5.671e-09  5.470e-10  10.369  < 2e-16 ***
price_levelprice_2  2.162e-01  2.075e-02  10.419  < 2e-16 ***
price_levelprice_3  2.113e-01  5.831e-02   3.624 0.000292 ***
price_levelprice_4  3.990e-01  9.271e-02   4.303 1.71e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7589 on 5959 degrees of freedom
Multiple R-squared:  0.08054,   Adjusted R-squared:  0.07962
F-statistic:    87 on 6 and 5959 DF,  p-value: < 2.2e-16
```



Q7 Keep the number of elite users and the price levels as the only predictors on average star ratings in the model. Add an appropriate transformation to the model to allow for a U-shaped effect from the number of elite users on average star rating. Use at least two additional models (I suggest estimating log and quadratic models).

Cubic Model:

```
Call:
lm(formula = biz.stars ~ elite_cnt + I(elite_cnt^2) + I(elite_cnt^3) +
    price_level, data = biz)

Residuals:
     Min       1Q   Median       3Q      Max
-2.42984 -0.47711  0.05344  0.54236  1.96495

Coefficients:
                    Estimate Std. Error t value Pr(>|t|)
(Intercept)        3.200e+00  1.465e-02 218.482  < 2e-16 ***
elite_cnt          6.943e-03  4.353e-04  15.949  < 2e-16 ***
I(elite_cnt^2)    -1.517e-05  1.297e-06 -11.694  < 2e-16 ***
I(elite_cnt^3)     5.671e-09  5.470e-10  10.369  < 2e-16 ***
price_levelprice_2 2.162e-01  2.075e-02  10.419  < 2e-16 ***
price_levelprice_3 2.113e-01  5.831e-02   3.624 0.000292 ***
price_levelprice_4 3.990e-01  9.271e-02   4.303 1.71e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.7589 on 5959 degrees of freedom
Multiple R-squared:  0.08054,   Adjusted R-squared:  0.07962
F-statistic:    87 on 6 and 5959 DF,  p-value: < 2.2e-16
```



Q8 What potential modeling challenges might arise if we include these two additional variables *(1) total review volume (biz.rws.cnt)* and *2) repeat customer frequency (repeated_cnt)*? Using appropriate diagnostic tests, demonstrate whether these concerns are real and severe.

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age +
    biz.rws.cnt + repeated_cnt, data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0121 -0.4640  0.0827  0.5518  1.8102

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.379e+00  3.299e-02 102.403  < 2e-16 ***
elite_cnt          -2.915e-03  5.339e-04  -5.459 4.97e-08 ***
price_levelprice_2  2.006e-01  2.058e-02   9.750  < 2e-16 ***
price_levelprice_3  2.006e-01  5.780e-02   3.471 0.000523 ***
price_levelprice_4  5.269e-01  9.165e-02   5.749 9.45e-09 ***
metroPhoenix_area   5.693e-02  3.067e-02   1.856 0.063456 .
metroPittsburgh     1.496e-01  4.066e-02   3.679 0.000237 ***
metroVegas_area     4.100e-02  3.205e-02   1.279 0.200833
biz_age            -9.472e-05  7.347e-06 -12.891  < 2e-16 ***
biz.rws.cnt         7.904e-04  1.298e-04   6.091 1.19e-09 ***
repeated_cnt        8.217e-03  2.233e-03   3.680 0.000235 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7511 on 5955 degrees of freedom
Multiple R-squared:  0.1001,    Adjusted R-squared:  0.09861
F-statistic: 66.25 on 10 and 5955 DF,  p-value: < 2.2e-16
```

Concern 1: Multicollinearity

- New variables may correlate with existing predictors (e.g., elite users may drive review volume).
- **Test: Variance Inflation Factor (VIF)**

```
> vif(lm_1)
                        GVIF Df GVIF^(1/(2*Df))
elite_cnt           1.154754  1        1.074595
factor(price_level) 1.076600  3        1.012377
factor(metro)       1.031231  3        1.005139
biz_age             1.082064  1        1.040223
> vif(lm_2)
                  GVIF Df GVIF^(1/(2*Df))
elite_cnt     8.523774  1        2.919550
price_level   1.114984  3        1.018306
metro         1.075077  3        1.012139
biz_age       1.095008  1        1.046426
biz.rws.cnt  15.590018  1        3.948420
repeated_cnt  5.790759  1        2.406400
```

- **Result: elite_cnt** and **biz.rws.cnt** have high VIF (>5), so there is multicollinearity between them

Q8 What potential modeling challenges might arise if we include these two additional variables *(1) total review volume (biz.rws.cnt)* and *2) repeat customer frequency (repeated_cnt)*? Using appropriate diagnostic tests, demonstrate whether these concerns are real and severe.

Concern 1: Multicollinearity → Real & Severe

- **elite_cnt** and **biz.rws.cnt** have high VIF (>5), so there is high multicollinearity between them
- **Solution: Drop One of the Collinear Variables**

```
> vif(lm_2_elite)
                GVIF Df GVIF^(1/(2*Df))
elite_cnt   3.041593  1        1.744016
price_level 1.110032  3        1.017550
metro       1.057417  3        1.009348
biz_age     1.093878  1        1.045886
repeated_cnt 3.085230 1        1.756482
> vif(lm_2_rws)
                GVIF Df GVIF^(1/(2*Df))
biz.rws.cnt 5.563086  1        2.358620
price_level 1.103582  3        1.016562
metro       1.048002  3        1.007845
biz_age     1.088987  1        1.043545
repeated_cnt 5.547987 1        2.355417
```

- **Conclusion: Dropping biz.rws.cnt and keeping elite_cnt gives smaller VIF → better option to deal with the multicollinearity**

Option A: Keep elite_cnt Drop biz.rws.cnt

```
Call:
lm(formula = biz.stars ~ elite_cnt + price_level + metro + biz_age +
    repeated_cnt, data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.0068 -0.4674  0.0795  0.5571  1.8076

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.379e+00  3.309e-02 102.114  < 2e-16 ***
elite_cnt          -3.065e-04  3.199e-04  -0.958 0.337996
price_levelprice_2  2.089e-01  2.059e-02  10.144  < 2e-16 ***
price_levelprice_3  2.081e-01  5.796e-02   3.591 0.000332 ***
price_levelprice_4  5.302e-01  9.193e-02   5.767 8.45e-09 ***
metroPhoenix_area   6.983e-02  3.069e-02   2.276 0.022891 *
metroPittsburgh     1.427e-01  4.077e-02   3.499 0.000470 ***
metroVegas_area     5.771e-02  3.203e-02   1.802 0.071636 .
biz_age            -9.616e-05  7.366e-06 -13.054  < 2e-16 ***
repeated_cnt        1.751e-02  1.635e-03  10.714  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7533 on 5956 degrees of freedom
Multiple R-squared:  0.09451,	Adjusted R-squared:  0.09314
F-statistic: 69.07 on 9 and 5956 DF,  p-value: < 2.2e-16
```

Option B: Keep biz.rws.cnt Drop elite_cnt

```
Call:
lm(formula = biz.stars ~ biz.rws.cnt + price_level + metro +
    biz_age + repeated_cnt, data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.9852 -0.4604  0.0824  0.5542  1.8138

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)         3.379e+00  3.307e-02 102.178  < 2e-16 ***
biz.rws.cnt         2.223e-04  7.770e-05   2.861 0.004237 **
price_levelprice_2  2.076e-01  2.059e-02  10.086  < 2e-16 ***
price_levelprice_3  1.866e-01  5.788e-02   3.223 0.001274 **
price_levelprice_4  5.023e-01  9.176e-02   5.474 4.58e-08 ***
metroPhoenix_area   7.484e-02  3.056e-02   2.449 0.014369 *
metroPittsburgh     1.420e-01  4.074e-02   3.487 0.000492 ***
metroVegas_area     5.475e-02  3.203e-02   1.709 0.087450 .
biz_age            -9.769e-05  7.345e-06 -13.301  < 2e-16 ***
repeated_cnt        1.071e-02  2.191e-03   4.890 1.03e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.7529 on 5956 degrees of freedom
Multiple R-squared:  0.09561,	Adjusted R-squared:  0.09425
F-statistic: 69.96 on 9 and 5956 DF,  p-value: < 2.2e-16
```

Q8 What potential modeling challenges might arise if we include these two additional variables *(1) total review volume (biz.rws.cnt)* and *2) repeat customer frequency (repeated_cnt)*? Using appropriate diagnostic tests, demonstrate whether these concerns are real and severe.

Concern 2: Omitted variable bias

- **Test:** Compare coefficients before/after adding variables.
- **Results:**
- **elite_cnt** coefficient flipped from +0.0024 (lm_1) to -0.0029 (lm_2).

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 3.354e+00  3.332e-02 100.649   < 2e-16 ***
elite_cnt   2.393e-03  1.990e-04  12.026   < 2e-16 ***
```

```
Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.379e+00  3.299e-02 102.403  < 2e-16 ***
elite_cnt   -2.915e-03  5.339e-04  -5.459 4.97e-08 ***
```

Conclusion:

- **In the initial model (lm_1), the effect of elite users was *confounded by omitted variables* (e.g., biz.rws.cnt), which inflated the estimate.**
- **After controlling for review volume and repeat customers in lm_2, elite users are associated with lower ratings, suggesting that the true effect of elite users may be critical or selective.**
- **Solution: Including other Instrumental variables (IV)**

Q8 What potential modeling challenges might arise if we include these two additional variables *(1) total review volume (biz.rws.cnt)* and *2) repeat customer frequency (repeated_cnt)*? Using appropriate diagnostic tests, demonstrate whether these concerns are real and severe.

Concern 3: Model Overfitting

- **Concern:** Added variables may improve fit artificially.
- **Test:** Compare adjusted R^2 and AIC&BIC of lm_1 vs. lm_2

- **Results:**
- **lm_1: Adj. R^2 = 0.0758**
- **lm_2: Adj. R^2 = 0.0986**

```
> AIC(lm_1, lm_2)
     df      AIC
lm_1 10 13674.90
lm_2 12 13527.95
> BIC(lm_1, lm_2)
     df      BIC
lm_1 10 13741.84
lm_2 12 13608.28
```

Conclusion:

- **Not model overfitting**
- **lm_2 improves fit (higher Adj. R^2, lower AIC & BIC), suggesting meaningful added explanatory power.**

Q9 Return to the regression model you built earlier (in part 1) and rerun it using bootstrap resampling with 1,000 iterations. Report the mean and confidence intervals of the coefficient of elite.cnt.

```
n=1000
for (i in 1:n) {
    #Creating a resampled dataset from the sample data
    sample_d = yelp[sample(1:nrow(yelp), nrow(yelp), replace = TRUE), ]
    #Running the regression on these data
    model_bootstrap <- lm(biz.stars ~ elite_cnt + factor(price_level)
                          + factor(metro) + biz_age, data=sample_d)
    ## summary(model_bootstrap)
    #Saving the coefficients
    sample_coef_intercept <-
        c(sample_coef_intercept, model_bootstrap$coefficients[1])

    sample_coef_elite_cnt <-
        c(sample_coef_elite_cnt, model_bootstrap$coefficients[2])

    sample_coef_p2 <-
        c(sample_coef_p2, model_bootstrap$coefficients[3])

    sample_coef_p3 <-
        c(sample_coef_p3, model_bootstrap$coefficients[4])

    sample_coef_p4 <-
        c(sample_coef_p4, model_bootstrap$coefficients[5])

    sample_coef_Phoenix <-
        c(sample_coef_Phoenix, model_bootstrap$coefficients[6])

    sample_coef_Pittsburgh <-
        c(sample_coef_Pittsburgh, model_bootstrap$coefficients[7])

    sample_coef_Vegas <-
        c(sample_coef_Vegas, model_bootstrap$coefficients[8])

    sample_coef_biz_age <-
        c(sample_coef_biz_age, model_bootstrap$coefficients[9])

}

coefs <- rbind(sample_coef_intercept, sample_coef_elite_cnt, sample_coef_p2,
               sample_coef_p3, sample_coef_p4, sample_coef_Phoenix,
               sample_coef_Pittsburgh, sample_coef_Vegas, sample_coef_biz_age)
```

```
coefs <- rbind(sample_coef_intercept, sample_coef_elite_cnt, sample_coef_p2,
               sample_coef_p3, sample_coef_p4, sample_coef_Phoenix,
               sample_coef_Pittsburgh, sample_coef_Vegas, sample_coef_biz_age)

## t() transposes the matrix
coefs_df=t(as.data.frame(coefs))
coefs_df=as.data.frame(coefs_df)
```

```
> library(dplyr)
> alpha=0.05
> coefs_df %>%
+     dplyr::summarize(mean = mean(sample_coef_elite_cnt),
+                      lower = mean(sample_coef_elite_cnt) - qt(1- alpha/2, (n() - 1))*sd(sample_coef_elite_cnt)/sqrt(n()),
+                      upper = mean(sample_coef_elite_cnt) + qt(1- alpha/2, (n() - 1))*sd(sample_coef_elite_cnt)/sqrt(n()))
         mean       lower       upper
1 0.002582083 0.002538082 0.002626083
>
>
> ##Use linear regression as a short cut to calcualte CI
> # Calculate the mean and standard error
> l.model <- lm(sample_coef_elite_cnt ~ 1, coefs_df)
>
> # Calculate the CI
> confint(l.model, level=0.95)
                  2.5 %      97.5 %
(Intercept) 0.002538082 0.002626083
```

mean: 0.002582083

95%CI: [0.002538082, 0.002626083]

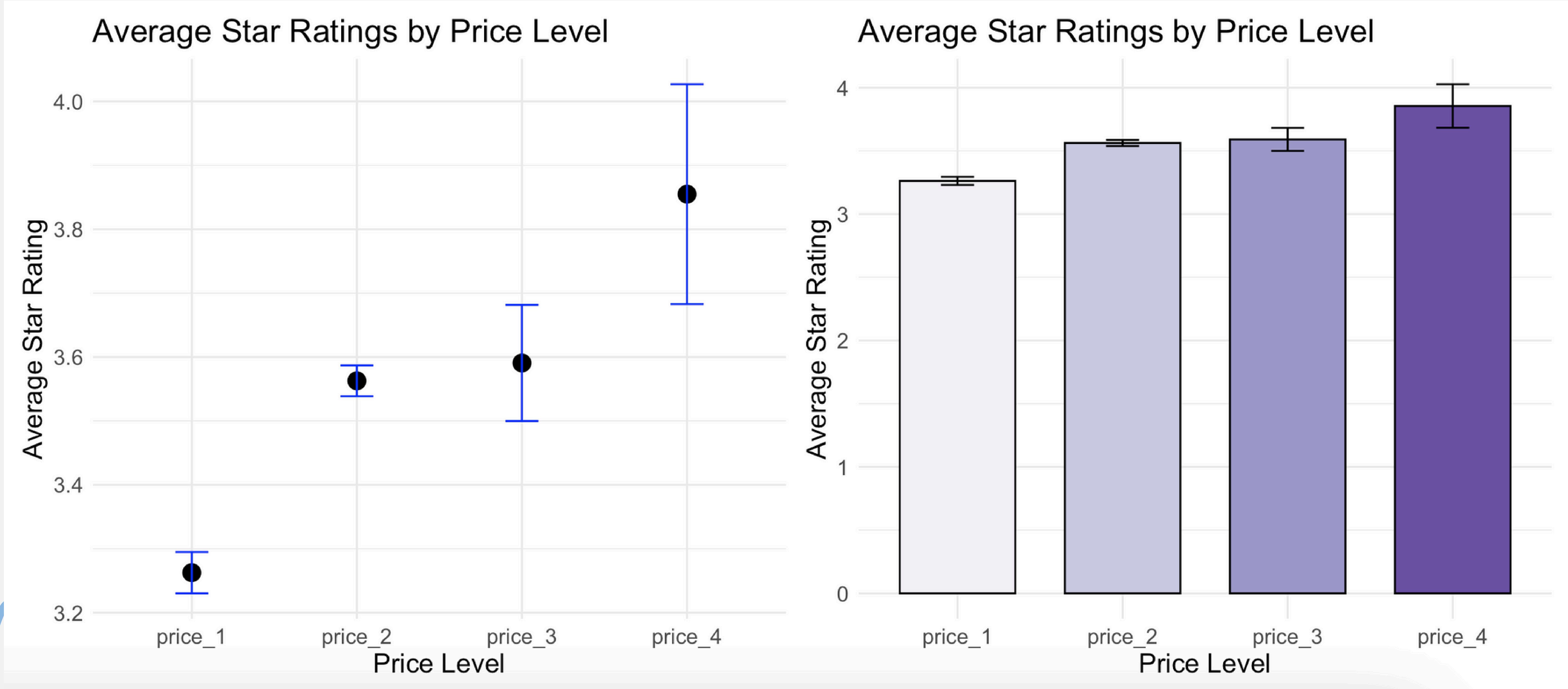
Q10 Compare the average star ratings across different price levels directly, using descriptive statistics and confidence intervals without relying on regression or other modeling assumptions. Report the output and visualize the differences with a plot of average star ratings (y axis) by price levels (x axis).

	price_level	n	mean_star	sd_star	se_star	t_val	ci_lower	ci_upper
	<chr>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	price_1	2887	3.26	0.886	0.0165	1.96	3.23	3.29
2	price_2	2823	3.56	0.655	0.0123	1.96	3.54	3.59
3	price_3	186	3.59	0.629	0.0461	1.97	3.50	3.68
4	price_4	70	3.85	0.721	0.0862	1.99	3.68	4.03

Trend: Higher price levels have higher mean star ratings.

- price_1 has the lowest mean rating (3.26).
- price_2 (3.56) and price_3 (3.59) have fairly close means and overlapping confidence intervals. Further testing is needed to tell if there’s a significant difference between them.
- price_4 (3.85) has the highest mean rating and a wider interval. Its range does not fully overlap with the lower groups’ intervals, suggesting it is indeed higher on average.

Q10 **Compare the average star ratings across different price levels directly, using descriptive statistics and confidence intervals without relying on regression or other modeling assumptions. Report the output and visualize the differences with a plot of average star ratings (y axis) by price levels (x axis).**



HW3

Group 9: Grace Chen, Vanessa Chen, Amanda Lee, Sheryl Xu

Business Question

Yelp Data Set
Specializing Influencing Marketing
Data Size: 5966 restaurants
Four Metropolitan Areas

Part 1: Rental Cost Analysis

Regression 1: rental_cost on dist_destination

```
Call:
lm(formula = rental_cost ~ dist_destination, data = biz)

Residuals:
    Min      1Q  Median      3Q     Max 
-4.5639 -0.9157 -0.0043  0.9105  5.2672 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)      57.533414   0.276581   208.0   <2e-16 ***
dist_destination -0.541653   0.005344  -101.4   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.329 on 5964 degrees of freedom
Multiple R-squared:  0.6327,	Adjusted R-squared:  0.6327 
F-statistic: 1.028e+04 on 1 and 5964 DF,  p-value: < 2.2e-16
```

Observation:

- Distance to the nearest tourist hotspot has a p-value < 5%, which means it’s a significant predictor to rental cost
- With every 1 mile increase in dist_destination, rental cost decreases by 0.54
- The model has a p-value < 5%, meaning this is a model with significant predicting power over rental cost

Part 1: Rental Cost Analysis

Regression 2: rental_cost on dist_destination + prime_location

```
Call:
lm(formula = rental_cost ~ dist_destination + factor(prime_location),
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
-4.1322 -0.6886  0.0138  0.6689  3.8873

Coefficients:
                        Estimate Std. Error t value Pr(>|t|)
(Intercept)            27.329091   0.503229  54.308   <2e-16 ***
dist_destination        0.012338   0.009318   1.324    0.185
factor(prime_location)1 4.064373   0.061526  66.059   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.01 on 5963 degrees of freedom
Multiple R-squared:  0.7879,	Adjusted R-squared:  0.7879
F-statistic: 1.108e+04 on 2 and 5963 DF,  p-value: < 2.2e-16
```

Observation:

- dist_destination is no longer a significant predictor, and coefficient changes from -0.54 to 0.012
- Prime_location is a significant predictor (p-value < 5%)
- Businesses in prime location pays $4.064 more rental than those not in prime location
- The model has a p-value < 5%, meaning this is a model with significant predicting power over rental cost

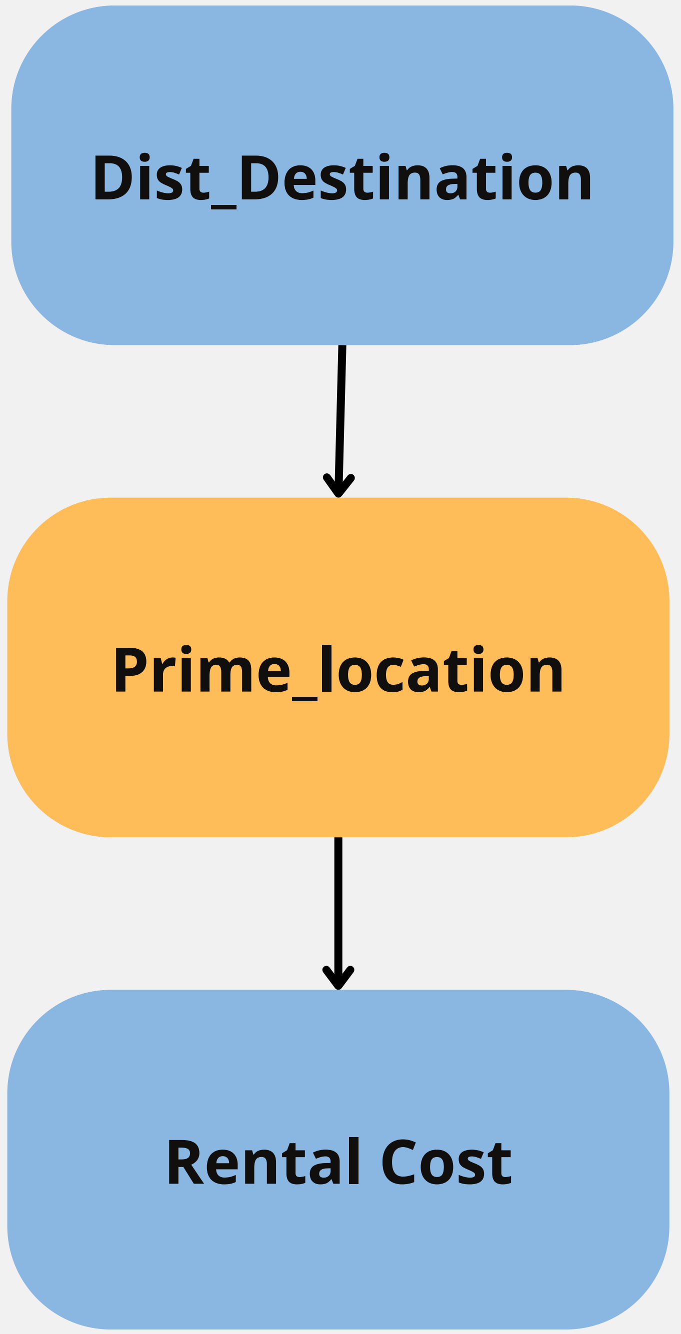
Part 1: Rental Cost Analysis

VIF on Regression 2 & DAG

```
> vif(lm_2)
     dist_destination factor(prime_location)
             5.264927               5.264927
```

VIF result shows **moderate multicollinearity** (>5) between destination to the nearest tourist hotspot and prime location. We proceed to examine the relationship between the rental cost and these two variables.

Pipe Structure



A **pipe structure** explains the confounding relationship and the change in dist_destination's effect after considering prime_location.

The true predictor of rental cost is prime location; however, distance to tourist hotspot contributes to making a restaurant location a prime location. Therefore, **adding prime_location removes the false effect of dist_destination** and reveals the real predictor.

Part 1: Rental Cost Analysis

Using drop1() to choose predictors

```
Model:
rental_cost ~ dist_destination + factor(prime_location)
                       Df Sum of Sq     RSS    AIC
<none>                               6085.4  124.2
dist_destination        1       1.8  6087.2  124.0
factor(prime_location)  1    4453.4 10538.8 3398.6
```

Drop1() results show that dropping prime_location variable will increase residual R-squared (RSS) and sum-squared.

This means **dropping prime_location will decrease model predictability because there will be more unexplained variance**.

Therefore, prime location is a strong predictor of rental cost.

Part 2: Health Inspection Analysis

Regression 1: inspector_visit & dist_destination

```
Call:
lm(formula = inspector_visit ~ dist_destination, data = biz)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.5002 -0.4907 -0.4657  0.5278  3.5370 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)       4.656911   0.220373  21.132   <2e-16 ***
dist_destination -0.003414   0.004258  -0.802    0.423    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.059 on 5964 degrees of freedom
Multiple R-squared:  0.0001078,	Adjusted R-squared:  -5.987e-05 
F-statistic: 0.6429 on 1 and 5964 DF,  p-value: 0.4227
```

Observation:

- Since the p-value of the dis-destination is greater than 0.05, we can't reject the null hypothesis.
- there's no sgnificant relationship between dist_destination and inspector visit.

Part 2: Health Inspection Analysis

Regression 2: inspector_visit & dist_destination + health_alarm

inspector_visit = 0.788 + 0.068dist_destination + 1.061 health_alarm

```
Call:
lm(formula = inspector_visit ~ dist_destination + factor(health_alarm),
    data = biz)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.5605 -0.5501 -0.0894  0.6379  3.6224 

Coefficients:
                      Estimate Std. Error t value Pr(>|t|)    
(Intercept)           0.788295   0.270471   2.915  0.00358 ** 
dist_destination      0.068375   0.005146  13.287  < 2e-16 ***
factor(health_alarm)1 1.060957   0.046308  22.911  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.015 on 5963 degrees of freedom
Multiple R-squared:  0.08101,	Adjusted R-squared:  0.0807 
F-statistic: 262.8 on 2 and 5963 DF,  p-value: < 2.2e-16
```

Observation:

- After introducing the factor of health_alarm, both dist_destination and health_alarm have a p-value less than 0.05, which rejects the null hypothesis.
- When the distance of destinations increased by one mile, the inspector visit increased by 0.06.
- Every health alarm happened can cause the inspector visit increased 1.06.

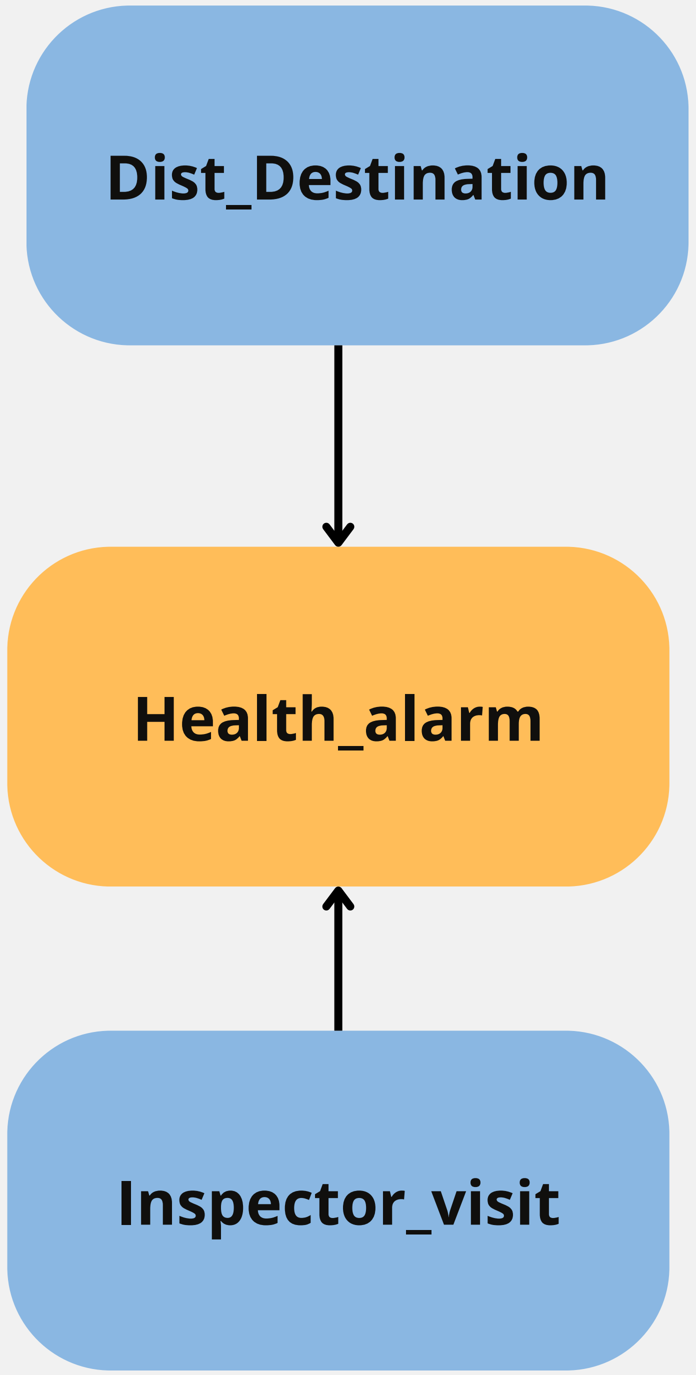
Part 2: Health Inspection Analysis

VIF and relationship

```
vif(lm_HI_2)
  dist_destination factor(health_alarm)
          1.589199             1.589199
```

VIF result shows **there's low to moderate multicollinearity** between destination to the nearest tourist hotspot and health alarm.
No further analysis needed

BUT...



The model is considered a **collider** because of the following factors:

- **dist_destination** is not correlated with inspector visits individually (in model 1, the p-value is greater than 0.05)
- After introducing **health_alarm** into the model, both **health_alarm** and **dist_destination** become significant (see model 2)
    - This is because of the **collider bias**.

*Thus, we should **control** health_alarm if we are examining the relationship between inspector visits and the destination to the nearest tourist hotspot. It artificially creates a false association between dist_destination and inspector_visit (even if none exists).*

Part 3: What Makes a Restaurant Popular?

```
Call:
lm(formula = biz.rws.cnt ~ rst.stars + factor(is_open) + rst.stars *
    factor(is_open), data = biz)

Residuals:
    Min      1Q  Median      3Q     Max
 -334.9   -97.5   -44.7    16.5 10175.4

Coefficients:
                            Estimate Std. Error t value Pr(>|t|)
(Intercept)                  -19.551     29.303  -0.667 0.504655
rst.stars                     25.773      8.334   3.092 0.001995 **
factor(is_open)1            -123.789     34.930  -3.544 0.000397 ***
rst.stars:factor(is_open)1    70.470      9.936   7.092 1.47e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 283.1 on 5962 degrees of freedom
Multiple R-squared:  0.08532,    Adjusted R-squared:  0.08485
F-statistic: 185.4 on 3 and 5962 DF,  p-value: < 2.2e-16
```

Main effects:

- For closed restaurant, each additional star increases the number of reviews by 25.773.
- For restaurants that are open, the number of reviews is expected to be 123.789 fewer than for closed restaurants.

Interaction:

- For every additional star, the increase in the number of reviews for open restaurants is 70.47 more than the increase for closed restaurants.

Reputation matters more in restoring popularity.

Part 3: What Makes a Restaurant Popular?

```
> em.pop_4_stars
 is_open emmean   SE   df lower.CL upper.CL
       0   83.5 7.69 5962     68.5     98.6
       1  241.6 5.58 5962    230.7    252.6

Confidence level used: 0.95
> pairs(em.pop_4_stars)
 contrast            estimate  SE   df t.ratio p.value
 is_open0 - is_open1     -158 9.5 5962 -16.634  <.0001
```

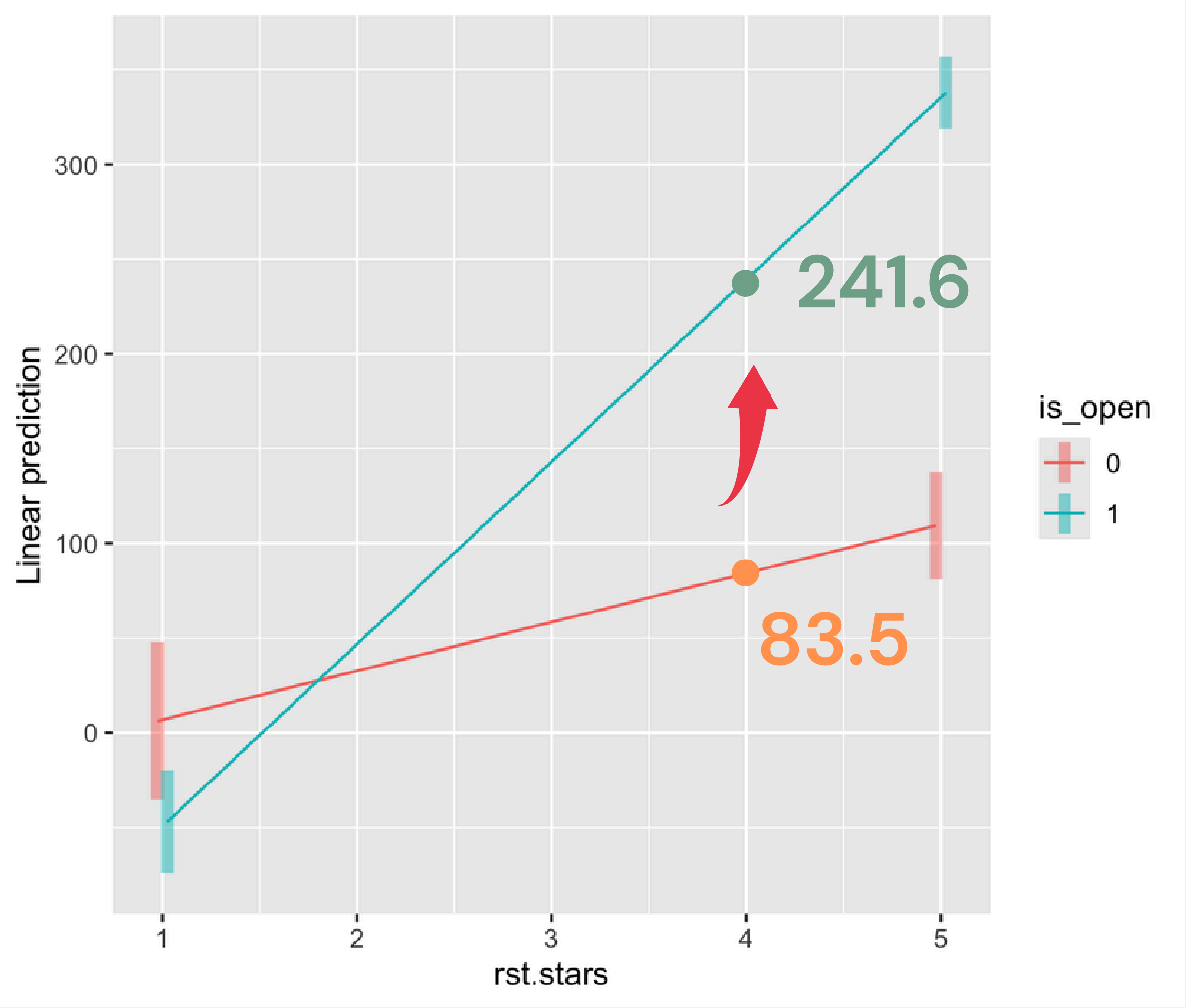
Mean popularity (reviews) for 4-star restaurants

- Closed: 83.5
- Open: 241.6
- Difference: 158

Open restaurants with 4 stars have 158 more reviews than closed restaurants with the same star rating on the average.

Visualization of the effect

```
> emmip(pop_interaction, factor(is_open) ~ rst.stars, CIs=TRUE,
+       at=list(rst.stars = c(1,5)))
```



Part 3: What Makes a Restaurant Popular?

```
> three_interaction = lm(biz.rws.cnt ~ elite_cnt* rst.stars* factor(is_open), data = biz)
> summary(three_interaction)

Call:
lm(formula = biz.rws.cnt ~ elite_cnt * rst.stars * factor(is_open),
    data = biz)

Residuals:
     Min       1Q   Median       3Q      Max
-1356.30   -35.23   -14.77    15.17  1810.30

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         -6.07582   10.96217  -0.554 0.579427
elite_cnt                            3.68712    0.63983   5.763  8.7e-09 ***
rst.stars                            8.02512    3.13241   2.562 0.010432 *
factor(is_open)1                    -9.84998   12.98768  -0.758 0.448236
elite_cnt:rst.stars                  0.08191    0.17152   0.478 0.632971
elite_cnt:factor(is_open)1          -1.21042    0.66575  -1.818 0.069094 .
rst.stars:factor(is_open)1          11.82286    3.72197   3.177 0.001498 **
elite_cnt:rst.stars:factor(is_open)1 0.63453    0.17816   3.562 0.000372 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 98.57 on 5958 degrees of freedom
Multiple R-squared:  0.8892,	Adjusted R-squared:  0.889
F-statistic:  6828 on 7 and 5958 DF,  p-value: < 2.2e-16
```

Main effects:

- For each additional elite Yelper, the number of reviews increases by 3.687 for restaurants that are closed, assuming the restaurant has zero stars.
- For each additional star, the number of reviews increases by 8.025, assuming the restaurant is closed and there are no elite Yelpers.

Part 3: What Makes a Restaurant Popular?

```
> three_interaction = lm(biz.rws.cnt ~ elite_cnt* rst.stars* factor(is_open), data = biz)
> summary(three_interaction)

Call:
lm(formula = biz.rws.cnt ~ elite_cnt * rst.stars * factor(is_open),
    data = biz)

Residuals:
     Min       1Q   Median       3Q      Max
-1356.30   -35.23   -14.77    15.17  1810.30

Coefficients:
                                    Estimate Std. Error t value Pr(>|t|)
(Intercept)                         -6.07582   10.96217  -0.554 0.579427
elite_cnt                            3.68712    0.63983   5.763  8.7e-09 ***
rst.stars                            8.02512    3.13241   2.562 0.010432 *
factor(is_open)1                    -9.84998   12.98768  -0.758 0.448236
elite_cnt:rst.stars                  0.08191    0.17152   0.478 0.632971
elite_cnt:factor(is_open)1          -1.21042    0.66575  -1.818 0.069094 .
rst.stars:factor(is_open)1          11.82286    3.72197   3.177 0.001498 **
elite_cnt:rst.stars:factor(is_open)1  0.63453    0.17816   3.562 0.000372 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 98.57 on 5958 degrees of freedom
Multiple R-squared:  0.8892,	Adjusted R-squared:  0.889
F-statistic:  6828 on 7 and 5958 DF,  p-value: < 2.2e-16
```

Two-way Interaction:

- For open restaurants, each additional star increases the number of reviews by 11.82286 more than for closed restaurants.

Three-way Interaction:

- For open restaurants, the number of reviews increases by 0.63453 more for each additional elite Yelper, for each additional star.

Part 3: What Makes a Restaurant Popular?

The difference in popularity between the restaurants **in good operation status** with **100 elite reviews** but with **average star rating of 4 and 5**

```
> em_4.5 <- emmeans(three_interaction, ~ rst.stars | elite_cnt + is_open,
+              at = list(elite_cnt = 100, is_open = 1, rst.stars = c(4, 5)))
> em_4.5
elite_cnt = 100, is_open = 1:
 rst.stars emmean   SE   df lower.CL upper.CL
         4    598 2.69 5958      592      603
         5    689 6.13 5958      677      701

Confidence level used: 0.95
> pairs(em_4.5)  # Calculates the difference
elite_cnt = 100, is_open = 1:
 contrast                estimate   SE   df t.ratio p.value
 rst.stars4 - rst.stars5    -91.5 4.62 5958 -19.810  <.0001
```

The emmeans function shows the mean popularity (biz.rws.cnt) when:

- **with 100 elite reviews: elite_cnt = 100**
- **in good operation status: is_open = 1**
- **average star rating of 4 and 5: rst.stars = c(4, 5)**

→ The mean review counts of 4-star restaurants is 598 and 689 for 5-star restaurants.

→ The difference in popularity (biz.rws.cnt) between 4-star and 5-star restaurants is 91.5.

Part 3: What Makes a Restaurant Popular?

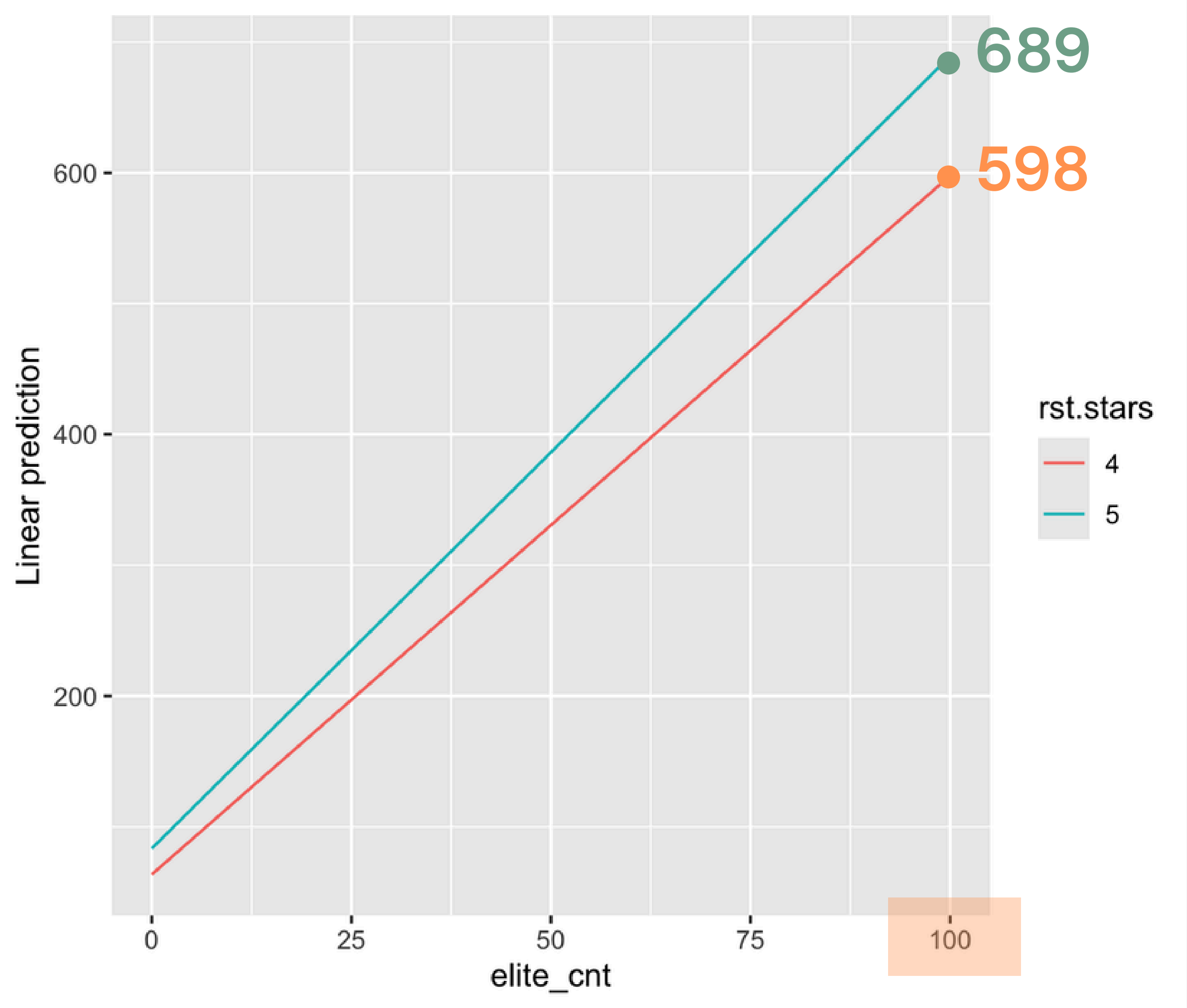
The difference in popularity between the restaurants **in good operation status** with **average star rating of 4 and 5**, **under different number of elite reviews**

```
# Create an emmGrid object
em_grid <- emmeans(three_interaction, ~ elite_cnt | rst.stars,
                   # Plot elite_cnt effects conditioned on star rating
    at = list(
        elite_cnt = seq(0, 100, by = 20),  # Range of elite Yelpers
        rst.stars = c(4, 5),                # Compare 4 vs. 5 stars
        is_open = 1                         # Only open restaurants
    )
)
```

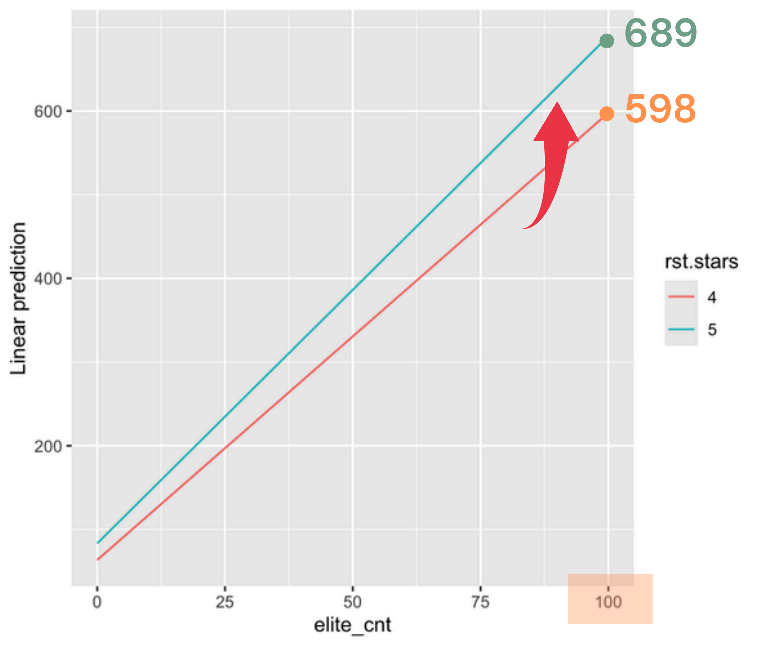
```
> em_grid
rst.stars = 4:
 elite_cnt emmean   SE   df lower.CL upper.CL
         0   63.5 2.15 5958     59.2     67.7
        20  170.3 1.98 5958    166.4    174.2
        40  277.2 1.95 5958    273.3    281.0
        60  384.0 2.08 5958    379.9    388.1
        80  490.9 2.34 5958    486.3    495.4
       100  597.7 2.69 5958    592.4    603.0

rst.stars = 5:
 elite_cnt emmean   SE   df lower.CL upper.CL
         0   83.3 3.73 5958     76.0     90.6
        20  204.5 3.47 5958    197.7    211.3
        40  325.7 3.67 5958    318.5    332.9
        60  446.8 4.27 5958    438.5    455.2
        80  568.0 5.12 5958    558.0    578.1
       100  689.2 6.13 5958    677.2    701.2

Confidence level used: 0.95
```

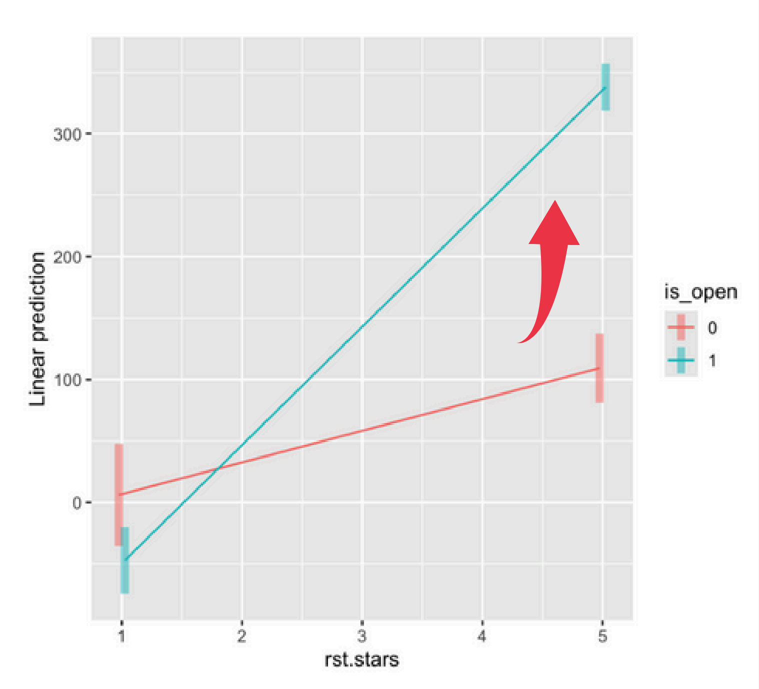
Visualization of the effect

```
> emmip(em_grid, rst.stars ~ elite_cnt)
```



Part 3: What Makes a Restaurant Popular?





Key Implications:

- **Reputation & Open Status Synergy:** High-rated (4–5 star) **open** restaurants gain significantly more reviews, especially when combined with **elite Yelpers**. The interaction between stars and operational status is critical—closed restaurants see minimal benefits from reputation alone.
- **Elite Yelper Amplification:** Elite influencers disproportionately boost popularity for **open, high-rated** restaurants (e.g., 5-star restaurants with 100 elite reviews get ~90 more reviews than 4-star counterparts).

Business Suggestions:

- **Target Campaigns:** Offer perks (e.g., “Elite Dining Events”) to encourage 4-star venues to improve to 5-star and maximize review growth
- **Promote Star Ratings**: Encourage restaurants to improve ratings (e.g., service training), as each star increase drives about 8 more baseline reviews, and even more with elite Yelpers.
- **Operational Priority:** Highlight open status in marketing (e.g., "Now Open!") to leverage its interaction with reputation.

Thank you very much!

Appendix

Part 1: Rental Cost Analysis

```
lm_1 = lm(rental_cost ~ dist_destination, data=biz)
summary(lm_1)

lm_2 = lm(rental_cost ~ dist_destination + factor(prime_location), data=biz)
summary(lm_2)

library(car)
vif(lm_1)
vif(lm_2)

anova(lm(rental_cost ~ dist_destination + factor(prime_location), biz))
anova(lm(rental_cost ~ factor(prime_location) + dist_destination, biz))
drop1(lm(rental_cost ~ dist_destination + factor(prime_location), biz))
```

Appendix

Part 2: Health Inspection Analysis

```
lm_HI_1 = lm(inspector_visit ~ dist_destination, data=biz)
summary(lm_HI_1)

lm_HI_2 = lm(inspector_visit ~ dist_destination + factor(health_alarm), data=biz)
summary(lm_HI_2)

vif(lm_HI_1)
vif(lm_HI_2)
```

Appendix

Part 3: What Makes a Restaurant Popular?

```
# Interaction effect
lm_pop = lm(biz.rws.cnt ~ rst.stars + factor(is_open) + rst.stars*factor(is_open), data=biz)
summary(lm_pop)

# Compare means of open vs. closed of 4-star restaurants
em.pop=emmeans(pop_interaction, "is_open")
em.pop_4_stars=emmeans(pop_interaction, "is_open", at = list(rst.stars = 4))
em.pop_4_stars
pairs(em.pop_4_stars)

## visualization
quantile(biz$rst.stars)
emmip(pop_interaction, factor(is_open) ~ rst.stars, CIs=TRUE,
    at=list(rst.stars = c(1,5)))
```

Appendix

Part 3: What Makes a Restaurant Popular?

```
# three_way_interaction
three_interaction = lm(biz.rws.cnt ~ elite_cnt* rst.stars* factor(is_open), data = biz)
summary(three_interaction)

# Compare differences
library(emmeans)
em_4.5 <- emmeans(three_interaction, ~ rst.stars | elite_cnt + is_open,
          at = list(elite_cnt = 100, is_open = 1, rst.stars = c(4, 5)))
em_4.5
pairs(em_4.5)  # Calculates the difference

## visualization
em_grid <- emmeans(three_interaction, ~ elite_cnt | rst.stars,
              # Plot elite_cnt effects conditioned on star rating
  at = list(
    elite_cnt = seq(0, 100, by = 20),  # Range of elite Yelpers
    rst.stars = c(4, 5),               # Compare 4 vs. 5 stars
    is_open = 1                        # Only open restaurants
  )
)
em_grid

# visualization
emmip(em_grid, rst.stars ~ elite_cnt)
```

HW4

Group 9: Grace Chen, Vanessa Chen, Amanda Lee, Sheryl Xu

Part 1: Factors Influencing Elite Review Attraction

1.1 Use binary logistic regression to predict has_elite (the presence of at least one elite review).

```
Call:
glm(formula = has_elite ~ rst.stars + price_level, family = binomial,
    data = biz)

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)          1.14889    0.17058   6.735 1.64e-11 ***
rst.stars            0.26793    0.05187   5.165 2.40e-07 ***
price_levelprice_2   0.23721    0.08981   2.641  0.00826 **
price_levelprice_3   0.58469    0.30487   1.918  0.05513 .
price_levelprice_4   0.63569    0.51985   1.223  0.22139
---
```

odds ratio:

```
> exp(coef(glm.model)[2])
rst.stars
 1.307256
> exp(coef(glm.model)[3])
price_levelprice_2
          1.267708
> exp(coef(glm.model)[4])
price_levelprice_3
          1.794428
> exp(coef(glm.model)[5])
price_levelprice_4
          1.888324
```

Interpretation:

- rst.stars: For each increase in restaurant stars，the log-odds of attracting an elite review increase by 0.26793 (odds ratio increases by 30.7%).
- price_level： compared to price level 1, price level 2's log-odds of attracting an elite review increase by 0.23 (odds ratio increases by 26.8%).

Part 1: Factors Influencing Elite Review Attraction

1.2 The likelihood of attracting elite reviews varies between different price levels.

```
> emmeans(glm.model, ~ price_level, type="response")
 price_level  prob      SE  df asymp.LCL asymp.UCL
 price_1     0.887 0.00600 Inf     0.875     0.899
 price_2     0.909 0.00548 Inf     0.898     0.919
 price_3     0.934 0.01840 Inf     0.887     0.962
 price_4     0.937 0.03040 Inf     0.844     0.976

Confidence level used: 0.95
Intervals are back-transformed from the logit scale
```

```
> pairs(emmeans(glm.model, ~ price_level, type="response"), reverse = T)
 contrast          odds.ratio    SE  df null z.ratio p.value
 price_2 / price_1       1.27 0.114 Inf    1   2.641  0.0411
 price_3 / price_1       1.79 0.547 Inf    1   1.918  0.2204
 price_3 / price_2       1.42 0.433 Inf    1   1.135  0.6676
 price_4 / price_1       1.89 0.982 Inf    1   1.223  0.6123
 price_4 / price_2       1.49 0.775 Inf    1   0.766  0.8697
 price_4 / price_3       1.05 0.627 Inf    1   0.086  0.9998

P value adjustment: tukey method for comparing a family of 4 estimates
Tests are performed on the log odds ratio scale
```

Interpretation:

- emmeans: Higher price level have slightly higher probabilities of receiving elite reviews, with price level 4 having the highest probability (0.937).
- price_2 / price_1: Odds ratio of 1.27, with a p-value < 0.05. Restaurants with price_2 have 27% higher odds of attracting elite reviews compared to price_1 (the cheapest tier), holding stars constant.
- All other pairs are not statistically significant, with p values all > 0.05.

Part 1: Factors Influencing Elite Review Attraction

1.3 restaurant's location and potential health concerns

```
Call:
glm(formula = has_elite ~ rst.stars + price_level + factor(prime_location) +
    dist_destination + factor(health_alarm), family = binomial,
    data = biz)

Coefficients:
                        Estimate Std. Error z value Pr(>|z|)
(Intercept)              0.13033    1.77543   0.073  0.94148
rst.stars                0.26804    0.05187   5.167 2.37e-07 ***
price_levelprice_2       0.23818    0.08984   2.651  0.00802 **
price_levelprice_3       0.58158    0.30497   1.907  0.05652 .
price_levelprice_4       0.63827    0.51992   1.228  0.21959
factor(prime_location)1  0.10811    0.20138   0.537  0.59135
dist_destination         0.01863    0.03272   0.569  0.56922
factor(health_alarm)1    0.09184    0.15315   0.600  0.54872
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 3941.8  on 5965  degrees of freedom
Residual deviance: 3895.4  on 5958  degrees of freedom
AIC: 3911.4

Number of Fisher Scoring iterations: 5
```

odds ratio

```
> exp(coef(glm.model2)[2])
rst.stars
   1.3074
> exp(coef(glm.model2)[3])
price_levelprice_2
          1.268937
> exp(coef(glm.model2)[4])
price_levelprice_3
          1.788857
> exp(coef(glm.model2)[5])
price_levelprice_4
          1.893207
```

```
> exp(coef(glm.model2)[4])
price_levelprice_3
          1.788857
> exp(coef(glm.model2)[5])
price_levelprice_4
          1.893207
> exp(coef(glm.model2)[6])
factor(prime_location)1
               1.114176
> exp(coef(glm.model2)[7])
dist_destination
        1.018801
> exp(coef(glm.model2)[8])
factor(health_alarm)1
             1.096193
```

Interpretation:

- rst.stars: for one increase in the restaurant's star rating, the log-odds of receiving an elite review increase by 0.26804 (odds ratio increase by 30.7%) compared to the price level 1. ($p<0.05$)
- price level 2: for restaurants with price level 2, the log-odds of receiving an elite review increase by 0.23818 (odds ratio increase by 26.8%) compared to the price level 1. ($p<0.05$)
- prime location, distance to a destination, health alarm: not significant ($p>0.05$), meaning no substantial impact on elite review likelihood.

Part 1: Factors Influencing Elite Review Attraction

1.4 Model 2 provides a better explanation of which restaurants attract elite reviews?

```
Likelihood ratio test

Model 1: has_elite ~ rst.stars + price_level
Model 2: has_elite ~ rst.stars + price_level + factor(prime_location) +
    dist_destination + factor(health_alarm)
  #Df  LogLik Df  Chisq Pr(>Chisq)
1   5 -1948.0
2   8 -1947.7  3 0.6384     0.8876
```

H0: The additional predictors do not improve the model fit. Model 1 is as good as Model 2.

H1: The additional predictors improve the model fit. Model 2 provides a better fit than Model 1.

Since p value > 0.05, we cannot reject H0. While Model 2 has a slightly higher log-likelihood (-1947.7 compared to -1948.0), the difference is very small. Therefore, we do not have enough evidence to say that adding the predictors improve the model fit.

Model 1 is as good as Model 2, so we recommend using Model 1 for predicting whether a restaurant attracts elite reviews because of its simplicity.

Part 2: Elite Reviews & Price Level

2.1 Pricing strategy and its affect on attracting elite reviewers.

	Dependent variable:		
	price_2	price_3	price_4
	(1)	(2)	(3)
factor(has_elite)1	0.327***	0.674**	0.803
	(0.088)	(0.304)	(0.518)
Constant	-0.315***	-3.356***	-4.454***
	(0.083)	(0.294)	(0.503)
Akaike Inf. Crit.	10,320.980	10,320.980	10,320.980
Note:	*p<0.1; **p<0.05; ***p<0.01		

Exponentiate coefficients yields RRR values.

```
> rrrs <- exp(coef(multinom.model))
> print(rrrs)
        (Intercept) factor(has_elite)1
price_2  0.72965116           1.386147
price_3  0.03488332           1.961486
price_4  0.01162797           2.231997
```

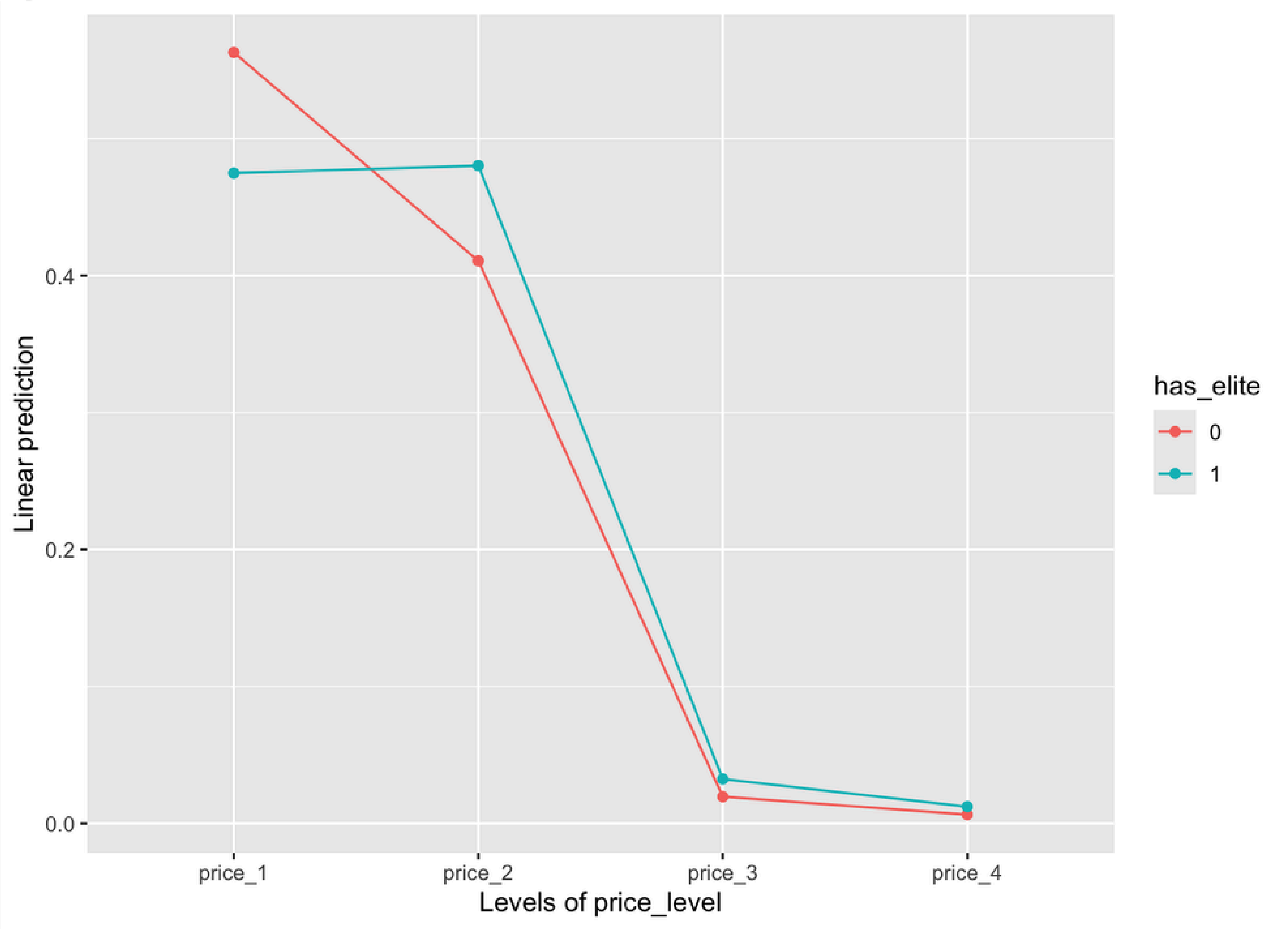
- Restaurants with elite reviews have 38.61% higher relative risk of being price-level 2 restaurants than price-level 1 restaurants.
- Restaurants with elite reviews have 96.15% higher relative risk of being price-level 3 restaurants than price-level 1 restaurants.
- Restaurants with elite reviews have 123.20% higher relative risk of being price-level 4 restaurants than price-level 1 restaurants; however, this difference is not statistically significicant.

Part 2: Elite Reviews & Price Level

2.2 The probability of being in each price level when there are elite reviews.

```
> #pairwise comparisons
> pairs(emmeans(multinom.model, ~ has_elite|price_level, mode="prob"))
price_level = price_1:
 contrast                estimate      SE df t.ratio p.value
 has_elite0 - has_elite1  0.08813 0.02120  6   4.158  0.0060

price_level = price_2:
 contrast                estimate      SE df t.ratio p.value
 has_elite0 - has_elite1 -0.06950 0.02100  6  -3.303  0.0164

price_level = price_3:
 contrast                estimate      SE df t.ratio p.value
 has_elite0 - has_elite1 -0.01285 0.00611  6  -2.102  0.0802

price_level = price_4:
 contrast                estimate      SE df t.ratio p.value
 has_elite0 - has_elite1 -0.00578 0.00359  6  -1.608  0.1590
```

```
> emmip(multinom.model, has_elite ~ price_level, mode="prob")
```



Restaurants with elite reviews are:

- 8.81% significantly less likely to be in price level 1.
- 6.95% significantly more likely to be in price level 2.
- 1.28% more likely to be in price level 3.
- 0.57% more likely to be in price level 4.

Having elite reviews most noticeably shifts the likelihood of restaurants being in price level 1 to price level 2, although there are increases in probability of being in price level 3 and 4, the difference is slight.

Part 2: Elite Reviews & Price Level

2.3 Business implications and marketing strategies

```
price_level = price_1:
 has_elite    prob      SE df lower.CL upper.CL
         0 0.56301 0.02010  6  0.51391   0.6121
         1 0.47488 0.00682  6  0.45819   0.4916

price_level = price_2:
 has_elite    prob      SE df lower.CL upper.CL
         0 0.41080 0.01990  6  0.36210   0.4595
         1 0.48030 0.00683  6  0.46359   0.4970

price_level = price_3:
 has_elite    prob      SE df lower.CL upper.CL
         0 0.01964 0.00561  6  0.00590   0.0334
         1 0.03249 0.00242  6  0.02656   0.0384

price_level = price_4:
 has_elite    prob      SE df lower.CL upper.CL
         0 0.00655 0.00326  6 -0.00144   0.0145
         1 0.01232 0.00151  6  0.00864   0.0160

Confidence level used: 0.95
```

The multinomial regression analysis shows that has_elite is a partially good predictor for price_level.

- Restaurants with elite reviews are less likely to be in the cheaper price range (price level 1), and more likely to be in the mid to high price ranges (price level 2 and 3).
- Price level 2 is the most probable level with elite reviews (48%).

Business implications for restaurants:

- Higher-range restaurants should leverage elite reviews in promotional content to signify quality and justify price range.
- If a restaurant is targeting a specific price range, attracting elite reviews can help them achieve their positioning.
- Premium restaurants (price level 4) may attract elite reviews, but lack of statistic significance suggest that they need extra branding and marketing effort to justify high price range.

Part 3: Further Exploration with Ordinal Logistic Regression

3.1 Ordinal logistic regression

```
> biz$price.f=as.factor(biz$price_level)
> biz$has_elite.f=as.factor(biz$has_elite)
> ordinal.model <- polr(price.f~ has_elite.f, data = biz, Hess = TRUE)
> summary(ordinal.model)
Call:
polr(formula = price.f ~ has_elite.f, data = biz, Hess = TRUE)

Coefficients:
              Value Std. Error t value
has_elite.f1 0.3644    0.08487   4.294

Intercepts:
                Value   Std. Error t value
price_1|price_2  0.2626  0.0806      3.2586
price_2|price_3  3.4369  0.1007     34.1196
price_3|price_4  4.7663  0.1433     33.2575

Residual Deviance: 10309.67
AIC: 10317.67
```

```
> #The summary function does not return p-values for the coefficients,
> #so we calculate them.
> ctable <- coef(summary(ordinal.model))
> p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
> ctable <- cbind(ctable, "p value" = p)
> ctable
                    Value Std. Error   t value       p value
has_elite.f1    0.3643964 0.08486986  4.293590  1.758066e-05
price_1|price_2 0.2625662 0.08057598  3.258616  1.119569e-03
price_2|price_3 3.4368802 0.10073035 34.119609 3.776602e-255
price_3|price_4 4.7662881 0.14331471 33.257494 1.590669e-242
```

Key coefficient for has_elite.f1 is:

- Estimate = 0.3644
- Standard Error = 0.08487
- p-value = 1.76e-05 (<0.05)

A positive coefficient ($\beta = 0.3644$) means elite-reviewed restaurants have:

- Lower odds of being in lower price levels (≤ price_1, ≤ price_2, etc.)
- Higher odds of being in higher price levels (≥ price_2, ≥ price_3, etc.)

$$\text{Odds Ratio} = e^{0.3644} \approx 1.44$$

Elite-reviewed restaurants have 44% higher odds (OR ≈ 1.44) of being in a higher price category than those without elite reviews, at every price threshold.

Part 3: Further Exploration with Ordinal Logistic Regression

3.2 Interpret the odds ratios of being at or below each price level (the intercepts)

```
> biz$price.f=as.factor(biz$price_level)
> biz$has_elite.f=as.factor(biz$has_elite)
> ordinal.model <- polr(price.f~ has_elite.f, data = biz, Hess = TRUE)
> summary(ordinal.model)
Call:
polr(formula = price.f ~ has_elite.f, data = biz, Hess = TRUE)

Coefficients:
              Value Std. Error t value
has_elite.f1 0.3644    0.08487   4.294

Intercepts:
                Value   Std. Error t value
price_1|price_2  0.2626  0.0806      3.2586
price_2|price_3  3.4369  0.1007     34.1196
price_3|price_4  4.7663  0.1433     33.2575

Residual Deviance: 10309.67
AIC: 10317.67
```

The high odd-ratio is because that the dataset **contains very few high-price-level restaurants** (e.g., price_3 and especially price_4), then: Most restaurants fall into price_1 or price_2, and
The model learns that the cumulative probability of being in a low price level is very high.

Threshold	Intercept (α_k)	Odds Ratio = $\exp(\alpha_k)$	Interpretation
price_1 | price_2	0.2626	≈ 1.30	Restaurants without elite reviews have **30% higher** odds of being price_1 or lower (vs. price_2 and above)
price_2 | price_3	3.4369	≈ 31.09	Restaurants without elite reviews have very **high odds (~ 31 times)** of being in price level 2 or below (vs. 3 or 4).
price_3 | price_4	4.7663	≈ 117.63	Restaurants without elite reviews **have extremely high odds (~ 118 times)** of being in price level 3 or below (vs. price level 4).

Part 3: Further Exploration with Ordinal Logistic Regression

3.3 the difference in predicted probabilities of being at price_level 4

```
> emmeans(ordinal.model, ~ has_elite.f|price.f, mode = "prob")
price.f = price_1:
 has_elite.f    prob      SE  df asymp.LCL asymp.UCL
 0           0.56527 0.01980 Inf   0.52646    0.6041
 1           0.47456 0.00681 Inf   0.46121    0.4879

price.f = price_2:
 has_elite.f    prob      SE  df asymp.LCL asymp.UCL
 0           0.40357 0.01760 Inf   0.36917    0.4380
 1           0.48118 0.00674 Inf   0.46797    0.4944

price.f = price_3:
 has_elite.f    prob      SE  df asymp.LCL asymp.UCL
 0           0.02272 0.00237 Inf   0.01808    0.0274
 1           0.03215 0.00233 Inf   0.02759    0.0367

price.f = price_4:
 has_elite.f    prob      SE  df asymp.LCL asymp.UCL
 0           0.00844 0.00120 Inf   0.00609    0.0108
 1           0.01211 0.00144 Inf   0.00928    0.0149

Confidence level used: 0.95
```

Difference in probabilities

=P(price_4 | has_elite=1)-P(price_4 | has_elite=0)

=0.01211-0.00844

=0.00367

Part 3: Further Exploration with Ordinal Logistic Regression

3.4 Comparison between ordinal and multinomial logistic regression

Price Level	Elite v.s. Non-elite	Multinomial Logistice Regression	Ordinal Logistice Regression	
price_1	Elite	0.47488	0.47456	elite lower than non-elite
	Non-Elite	0.56301	0.56527	
price_2	Elite	0.4803	0.48118	elite higher than non-elite
	Non-Elite	0.4108	0.40357	
price_3	Elite	0.03249	0.03215	elite slightly higher than non-elite
	Non-Elite	0.01964	0.02272	
price_4	Elite	0.01232	0.01211	elite slightly higher than non-elite
	Non-Elite	0.00655	0.00844	

Both ordinal and multinomial logistic regression models show consistent patterns in how elite reviews (has_elite) relate to restaurant price level:

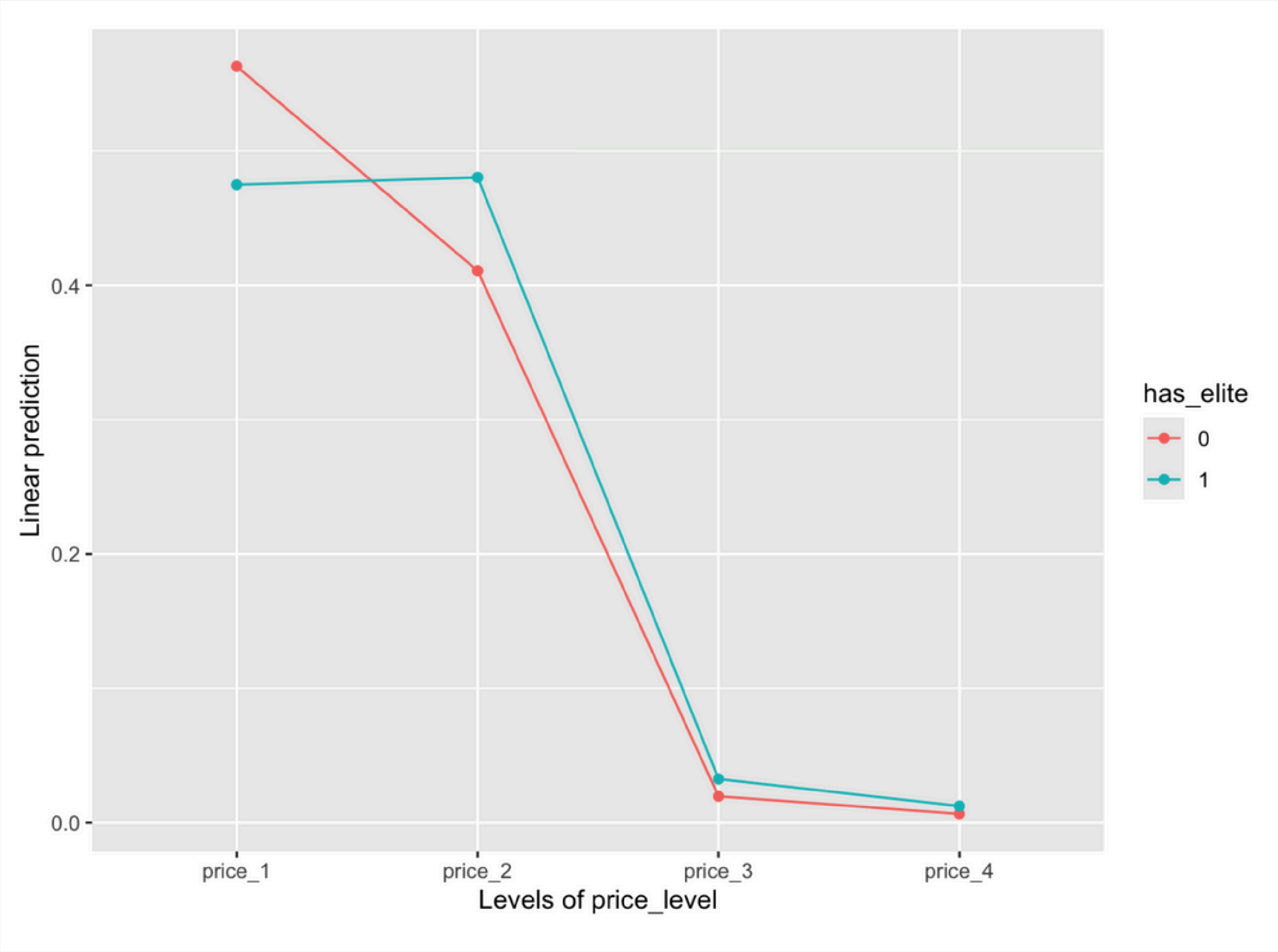
- Restaurants with elite reviews (has_elite = 1) are more likely to be in higher price categories, particularly price_2, price_3, and price_4, although the effect weakens for price level 4.

Part 3: Further Exploration with Ordinal Logistic Regression

3.4 Comparison between ordinal and multinomial logistic regression

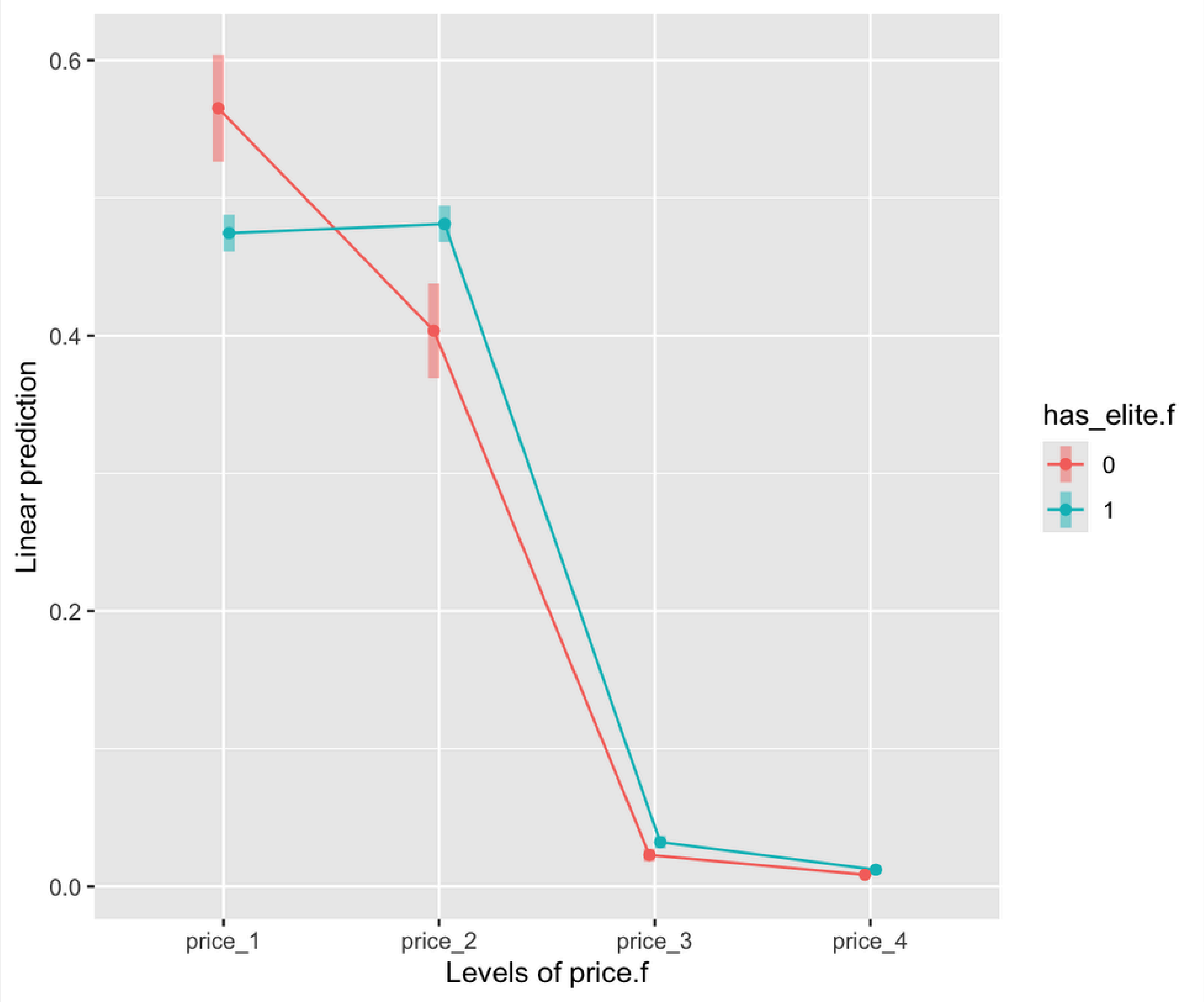
Multinomial logistic regression

```
> emmip(multinom.model, has_elite ~ price_level, mode="prob")
```



Ordinal logistic regression

```
> emmip(ordinal.model,  has_elite.f~ price.f, CIs = T, mode = "prob")
```



Part 3: Further Exploration with Ordinal Logistic Regression

3.4 Which model to choose?

Use the brant test is used to test the parallel regression assumption of ordinal logistic regression.

```
> brant(ordinal.model)
--------------------------------------------
Test for        X2      df      probability
--------------------------------------------
Omnibus         0.65    2       0.72
has_elite.f1    0.65    2       0.72
--------------------------------------------

H0: Parallel Regression Assumption holds
```

- **p-value = 0.72 > 0.05, we fail to reject the null hypothesis. This means the parallel regression assumption holds: the relationship between has_elite and the cumulative odds of being in a higher price category is consistent across thresholds.**

The ordinal logistic model is more appropriate:

- The **parallel regression assumption holds holds** (validated by the Brant test),
- **Price levels are naturally ordered** (e.g., price_1 < price_2 < price_3 < price_4
- **Ordinal model offers better interpretability and efficiency** (as shown earlier)

Part 3: Further Exploration with Ordinal Logistic Regression

Data Insight:

- Restaurants with elite reviews are significantly less likely to be in the cheapest tier (price_1) and more likely to occupy mid-to-high tiers (price_2–price_4).
- The largest shift occurs in price_2, where elite-reviewed restaurants have a 48% probability of appearing—making it the most common category for elite-affiliated venues.

Strategic Recommendations

01.

Yelp's Role:

- Enhance transparency by showing elite review distribution across price tiers and debunking the myth that elite feedback = high cost.
- Yelp might offer filters based on elite reviewer presence across **price tiers**, helping users find high-value options.

02.

Restaurant Actions:

- Attracting elite reviewers may **enhance reputation** but does **not strongly justify price increases**—especially in higher tiers where elite presence diminishes.
- Since elite reviewers are more active at lower price levels, affordable restaurants can leverage this by promoting elite feedback to drive traffic and trust.
- Mid-tier establishments (price_2) should actively encourage elite reviews (e.g., through exceptional service or loyalty programs) to capitalize on this demand shift.

Thank you very much!

Appendix

Part 1:

```
### Q1.1
biz$has_elite = ifelse(biz$elite_cnt > 0, 1, 0)

glm.model=glm(has_elite ~ rst.stars + price_level, data=biz, family=binomial)

summary(glm.model)

coef(glm.model)[2]

exp(coef(glm.model)[2])
exp(coef(glm.model)[3])
exp(coef(glm.model)[4])
exp(coef(glm.model)[5])

### Q1.2
library(emmeans)
emmeans(glm.model, ~ price_level)
emmeans(glm.model, ~ price_level, type="response")

pairs(emmeans(glm.model, ~ price_level, type="response"), reverse = T)

pairs(emmeans(glm.model2, ~ metro, type="response"), reverse = T)
```

```
### Q1.3
str(biz)
glm.model2=glm(has_elite ~ rst.stars + price_level + factor(prime_location)
 + dist_destination + factor(health_alarm), data=biz, family=binomial)
summary(glm.model2)
exp(coef(glm.model2)[6])
exp(coef(glm.model2)[7])
exp(coef(glm.model2)[8])

## Q1.4
## Likelihood ratio test (LRT) for model1 and model2
install.packages("lmtest")
library(lmtest)
lrtest(glm.model, glm.model2)
```

Appendix

Part 2:

```
## Q2

## multinomial logistic regression
#install.packages("nnet")
library(nnet)
multinom.model=multinom(price_level ~ factor(has_elite),
                        data=biz, maxit=1000)

summary(multinom.model)

install.packages("stargazer")
stargazer::stargazer(multinom.model, type = "text")

# Relative risk ratio (RRR): exponentiate the multinomial logit coefficients
rrrs <- exp(coef(multinom.model))
print(rrrs)


library(emmeans)
emmeans(multinom.model, ~ has_elite|price_level, mode="latent") #logit
emmeans(multinom.model, ~ has_elite|price_level, mode="prob") #probability

#pairwise comparisons
pairs(emmeans(multinom.model, ~ has_elite|price_level, mode="prob"))

emmip(multinom.model, has_elite ~ price_level, mode="prob") #probability
```

Appendix

Part 3:

```
## Q3
##Ordinal logistic regression
library(MASS)
biz$price.f=as.factor(biz$price_level)
biz$has_elite.f=as.factor(biz$has_elite)

ordinal.model <- polr(price.f~ has_elite.f, data = biz, Hess = TRUE)
summary(ordinal.model)

#The summary function does not return p-values for the coefficients,
#so we calculate them.
ctable <- coef(summary(ordinal.model))
p <- pnorm(abs(ctable[, "t value"]), lower.tail = FALSE) * 2
ctable <- cbind(ctable, "p value" = p)
ctable

emmeans(ordinal.model, ~ has_elite.f|price.f, mode = "prob")
emmip(ordinal.model,  has_elite.f~ price.f, CIs = T, mode = "prob")

emmeans(ordinal.model, ~ price.f|has_elite.f, mode = "prob")
emmip(ordinal.model,  has_elite.f~ price.f, CIs = T, mode = "prob")

##brant test
install.packages("brant")
library(brant)
brant(ordinal.model)
```

HW5

Group 9: Grace Chen, Vanessa Chen, Amanda Lee, Sheryl Xu

Part 1: Baseline Model

1.1 Elite reviews, price levels and effects on restaurant staying open

```
Call:
glm(formula = is_open ~ price_level + elite_cnt, family = binomial,
    data = biz)

Coefficients:
                    Estimate Std. Error z value Pr(>|z|)
(Intercept)         0.619096   0.041607  14.879  < 2e-16 ***
price_levelprice_2 -0.575534   0.058184  -9.892  < 2e-16 ***
price_levelprice_3 -1.128083   0.164466  -6.859 6.93e-12 ***
price_levelprice_4 -1.198810   0.271761  -4.411 1.03e-05 ***
elite_cnt           0.017377   0.001379  12.598  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7799.2  on 5965  degrees of freedom
Residual deviance: 7482.0  on 5961  degrees of freedom
AIC: 7492

Number of Fisher Scoring iterations: 5
```

Variable	Coefficient	Odds Ratio = exp(coef)
elite_cnt	0.017377	exp(0.017377)≈1.0175

Elite Status:

- **Each additional review increases the likelihood of a restaurant staying open by about 1.75% (exp^0.017).**

Part 1: Baseline Model

1.1 Elite reviews, price levels and effects on restaurant staying open

```
Call:
glm(formula = is_open ~ price_level + elite_cnt, family = binomial,
    data = biz)

Coefficients:
                   Estimate Std. Error z value Pr(>|z|)
(Intercept)        0.619096   0.041607  14.879  < 2e-16 ***
price_levelprice_2 -0.575534   0.058184  -9.892  < 2e-16 ***
price_levelprice_3 -1.128083   0.164466  -6.859 6.93e-12 ***
price_levelprice_4 -1.198810   0.271761  -4.411 1.03e-05 ***
elite_cnt          0.017377   0.001379  12.598  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 7799.2  on 5965  degrees of freedom
Residual deviance: 7482.0  on 5961  degrees of freedom
AIC: 7492

Number of Fisher Scoring iterations: 5
```

Price Level	Coefficient	Odds Ratio = exp(coef)	Interpretation
price_2	-0.575534	≈ 0.56	44% lower odds of staying open vs. price_1
price_3	-1.128083	≈ 0.32	68% lower odds of staying open vs. price_1
price_4	-1.19881	≈ 0.30	70% lower odds of staying open vs. price_1

Price Level:

- **Price level 2 restaurants are 44% (exp^-0.57) less likely to stay open than price level 1 restaurants.**
- **Price level 3 restaurants are 68% (exp^-1.12) less likely to stay open than price level 1 restaurants.**
- **Price level 4 restaurants are 70% (exp^-1.19) less likely to stay open than price level 1 restaurants.**

Part 1: Baseline Model

1.2 & 1.3 Setting threshold to minimize total expected cost

Costs of false prediction:

- Each False Positive prediction (predicting restaurant open when it's closed): $55
- Each False Negative prediction (predicting restaurant closed when it's open): $20

Cost-Sensitive Thresholding

- Calculating total expected costs at each threshold shows that Optimal Threshold that minimizes total expected cost is 0.7.

HOWEVER...

Sensitivity and specificity should be balanced to deliver good user experience.

Youden's Index

Optimal Youden's index that balances both sensitivity and specificity and minimizes cost is 0.574.

Confusion matrix at 0.574 as threshold:

Confusion Matrix		Actual	
		Closed	Open
Predicted	Closed	935	853
	Open	1215	2963

Part 1: Baseline Model

1.4 Mean accuracy across all thresholds between the cost-sensitive threshold and the threshold that maximizes Youden's Index.

```
> all_coords <- coords(roc_obj, "all", ret = c("threshold", "tp", "tn", "fp", "fn"))
>
> filtered_coords <- subset(all_coords, threshold >= 0.574 & threshold <= 0.7)
>
> filtered_coords$accuracy <- (filtered_coords$tp + filtered_coords$tn) /
+     (filtered_coords$tp + filtered_coords$tn + filtered_coords$fp + filtered_coords$fn)
>
> mean_accuracy <- mean(filtered_coords$accuracy)
>
> cat("Mean Accuracy between thresholds 0.574 and 0.7 is", round(mean_accuracy, 4), "\n")
Mean Accuracy between thresholds 0.574 and 0.7 is 0.574
```

Mean accuracy across all thresholds between the cost-sensitive threshold and the threshold that maximizes Youden's Index is 0.574.

In the range of thresholds, the model's predictions are correct about 57.4% of the time on average.

Part 1: Model 2, including average star rating and number of reviews from repeated consumers

1.5 Model 2's predictive power compared to Model 1

```
> print(roc_obj1)

Call:
roc.default(response = biz$is_open, predictor = pred_probs1)

Data: pred_probs1 in 2150 controls (biz$is_open 0) < 3816 cases (biz$is_open 1).
Area under the curve: 0.6456
> print(roc_obj2)

Call:
roc.default(response = biz$is_open, predictor = pred_probs2)

Data: pred_probs2 in 2150 controls (biz$is_open 0) < 3816 cases (biz$is_open 1).
Area under the curve: 0.703
```

Yelp's guideline:

- 0.6–0.7: Poor (weak predictive power).
- 0.7–0.8: Fair (moderate usefulness for decision-making).

Model 1

- **AUC: 0.6456**

Model 2:

- **AUC: 0.703**

Model 2 shows a meaningful improvement over Model 1, increasing the AUC from 0.6456 (poor predictive power) to 0.703 (fair predictive power).
Model 2 is more reliable and useful for decision-making.

Part 1: Model 2, including average star rating and number of reviews from repeated consumers

1.6 Identify high-probability restaurant segments

```
> print(gains_table)
```

Depth of File	N	Cume N	Mean Resp	Cume Mean Resp	Cume Pct of Total Resp	Lift Index	Cume Lift	Mean Model Score
10	596	596	0.88	0.88	13.7%	137	137	0.92
20	597	1193	0.84	0.86	26.8%	131	134	0.80
30	598	1791	0.80	0.84	39.4%	125	131	0.75
40	595	2386	0.70	0.81	50.4%	110	126	0.70
50	597	2983	0.65	0.77	60.5%	101	121	0.66
60	596	3579	0.59	0.74	69.7%	92	116	0.61
70	597	4176	0.60	0.72	79.0%	93	113	0.57
80	596	4772	0.56	0.70	87.7%	87	110	0.52
90	597	5369	0.44	0.67	94.6%	69	105	0.48
100	597	5966	0.35	0.64	100.0%	54	100	0.40

- segments where lift index (non-cumulative) > 100%: top 5 deciles (Depths 10–50)
- % of open restaurants captured in these high-lift segments: 60.5%
- % of total restaurants covered (to assess targeting efficiency): 2983/5966 = 50%

Part 1: Model 2, including average star rating and number of reviews from repeated consumers

1.6 Marketing strategy recommendations for Yelp

- Prioritize restaurants in the top 5 deciles of predicted survival probability for the delivery service pilot.
  - These segments achieve a Lift Index > 100%, meaning they significantly outperform random targeting. Specifically, by targeting just 50% of the restaurant population, Yelp can capture over 60% of open businesses, demonstrating high targeting efficiency.
  - Marketing efforts (e.g., promotional credits) should also be concentrated in these high-lift groups. Additionally, segment-specific messaging can be used to appeal to business owners' demonstrated strength and survival probability, reinforcing Yelp's value as a trusted partner.

Part 2.1: Model Evaluations with Test Set

2.1.1. Calculate and report the AUC of each model on both the train and test sets

```
set.seed(123)
train_indices <- sample(1:nrow(biz), size = 0.7 * nrow(biz))

train_data <- biz[train_indices, ]
test_data <- biz[-train_indices, ]
```

```
# Predictions
train_pred1 <- predict(glm.model1, train_data, type = "response")
test_pred1 <- predict(glm.model1, test_data, type = "response")

train_pred2 <- predict(glm.model2, train_data, type = "response")
test_pred2 <- predict(glm.model2, test_data, type = "response")
```

Step 1 – Partition:

- Randomly reserve 30% of data as the test set (holdout).

Step 2 – Train:

- Use the remaining 70% (training set) to build the model.

Step 3 – Evaluate:

- Test the model on the untouched test set to measure real‑world performance.

```
model1_auc <- c(
    auc(roc(train_data$is_open, train_pred1)),
    auc(roc(test_data$is_open, test_pred1)))

names(model1_auc) <- c("Train Set", "Test Set")
```

AUC of model 1 on train and test sets

```
> print(model1_auc)
Train Set  Test Set
0.6479150 0.6402141
```

```
model2_auc <- c(
    auc(roc(train_data$is_open, train_pred2)),
    auc(roc(test_data$is_open, test_pred2)))

names(model2_auc) <- c("Train Set", "Test Set")
```

AUC of model 2 on train and test sets

```
> print(model2_auc)
Train Set  Test Set
0.7052925 0.6974862
```

For both model: Train AUC ≈ Test AUC

→ Stable performance & minimal overfitting

→ Trust the model.

Part 2.1: Model Evaluations with Test Set

2.1.2. Which model would you recommend based on the AUC comparison?

Model	Train Set AUC	Test Set AUC	Predictive Power
Model 1	0.6479	0.6402	0.6–0.7: Poor
Model 2	0.7053	0.6975	0.7–0.8: Fair

Recommend Model 2:

Model 2 consistently outperforms Model 1 on both the training and test sets.

- Model 1's test set AUC falls in the "poor" range (0.6–0.7)
- Model 2's test set AUC is almost in the "fair" range (0.7–0.8), based on Yelp's guidelines.

- The test AUC improvement from 0.6402 to 0.6975 indicates that Model 2 generalizes better to unseen data.
- The training and test AUCs of Model 2 are close, suggesting low overfitting despite the more complex specification.

Part 2.2: Model 3

10-fold cross validation of three models

Model 3 = glm(is_open ~ poly(elite_cnt, 2, raw=T) + price_level*biz.stars*repeated_cnt + city)

Model	Mean AUC (Train)	Mean AUC (Test)
Model 1	0.6464	0.6373
Model 2	0.7035	0.6962
Model 3	0.7217	0.7122

- When the new Model 3 is introduced, we can see the average AUC is higher than the previous two models, which means Model 3 has a stronger prediction power.
- By checking the train AUC and test AUC, both Model 2 and Model 3 show a good ability of generalization. Model 1 showed a poor prediction power since the mean AUCs are similar to the baseline AUC.

Part 2.2: Model 3

AUC comparison and best model

Model	Mean AUC (Train)	Mean AUC (Test)	% Change compared to baseline model
Model 1	0.6464	0.6373	
Model 2	0.7035	0.6962	9.24% (marginal)
Model 3	0.7217	0.7122	11.75% (marginal)

**from model 2 to model 3: improved 1.82%*

- **Improvement Magnitude:**

Model 2 achieves a +9.24% AUC gain over Model 1, which crosses the 5% threshold for marginal improvement. Model 3, while the best performer, offers 11.75% improvement of AUC over Model 1. Still within the marginal range.

- **Complexity-Utility Tradeoff:**

Model 3 introduces significant complexity for a minimal performance gain. This risks overfitting in practice, especially if the sample size is limited. Model 2 provides a better balance: it captures meaningful patterns without excessive complexity.

- **Practical Considerations:**

If future data exhibits distribution shifts, Model 2's simplicity may generalize more robustly.

Appendix

Part 1:

```
### Q1.1
biz$has_elite = ifelse(biz$elite_cnt > 0, 1, 0)

glm.model=glm(has_elite ~ rst.stars + price_level, data=biz,
family=binomial)

summary(glm.model)

coef(glm.model)[2]

exp(coef(glm.model)[2])
exp(coef(glm.model)[3])
exp(coef(glm.model)[4])
exp(coef(glm.model)[5])
```

```
### Q1.2
# predictions from the model
pred_probs <- predict(glm.model1, type = "response")


install.packages("pROC") #if you have not installed the package
library(pROC)
# ROC curve analysis
roc_obj <- roc(biz$is_open, pred_probs)

plot(roc_obj, main = "ROC Curve", print.auc = T,
    legacy.axes = TRUE, lwd = 2)


#Cost-Sensitive Thresholding
cost_FP <- 55
cost_FN <- 20


library(dplyr)
# Calculate costs for all thresholds
# coords returns the coordinates of the ROC curve at one or several specified point(s).
costs <- coords(roc_obj, "all",
             ret = c("threshold", "fp", "fn")) %>%
  mutate(total_cost = fp * cost_FP + fn * cost_FN)

# count the fp and fn under each thresholld
# coords(roc_obj, "all", ret = c("threshold", "fp", "fn"))


# Find optimal threshold: minimizing total cost
optimal_threshold <- costs$threshold[which.min(costs$total_cost)]
print(paste("Optimal threshold:", round(optimal_threshold, 3)))
```

Appendix

Part 1:

```
### Q1.3
##** youden index: This threshold represents the point where the
model achieves **
# Extract sensitivity, specificity, and thresholds
youden <- coords(roc_obj, "all", ret = c("threshold", "sensitivity",
"specificity"))
youden


# Calculate Youden's J and find the optimal threshold
youden$youden_j <- youden$sensitivity + youden$specificity - 1
youden

optimal_idx <- which.max(youden$youden_j)
optimal_threshold2 <- youden$threshold[optimal_idx]

print(paste("Optimal Threshold (Youden's Index):",
round(optimal_threshold2, 3)))
```

```
# count the tp tn fp and fn under optimal_threshold2
coords(roc_obj, optimal_threshold2, ret = c("threshold", "tp", "tn", "fp", "fn"))

## matrix
conf_matrix <- matrix(
  c(2963, 1215, 853, 935),
  nrow = 2,
  byrow = TRUE,
  dimnames = list(
    "Predicted" = c("Positive", "Negative"),
    "Actual" = c("Positive", "Negative")
  )
)

print(conf_matrix)
```

Appendix

Part 1:

```
### Q1.4
# Step 1: Get all threshold-level stats from the ROC object
all_coords <- coords(roc_obj, "all", ret = c("threshold", "tp", "tn", "fp",
"fn"))

# Step 2: Filter thresholds between 0.574 and 0.7
filtered_coords <- subset(all_coords, threshold >= 0.574 &
threshold <= 0.7)

# Step 3: Calculate accuracy for each threshold
filtered_coords$accuracy <- (filtered_coords$tp +
filtered_coords$tn) /
   (filtered_coords$tp + filtered_coords$tn + filtered_coords$fp +
filtered_coords$fn)

# Step 4: Compute mean accuracy
mean_accuracy <- mean(filtered_coords$accuracy)

# Step 5: Print result
cat("Mean Accuracy between thresholds 0.574 and 0.7 is",
round(mean_accuracy, 4), "\n")
```

```
### Q1.5
glm.model2= glm(is_open ~ elite_cnt + price_level * biz.stars +
repeated_cnt, biz, family =
              binomial)

summary(glm.model2)

# model 1
pred_probs1 <- predict(glm.model1, type = "response")

library(pROC)
roc_obj1 <- roc(biz$is_open, pred_probs1)
print(roc_obj1)

plot(roc_obj1, main = "ROC Curve", print.auc = T,
    legacy.axes = TRUE, lwd = 2)

# model 2
pred_probs2 <- predict(glm.model2, type = "response")

roc_obj2 <- roc(biz$is_open, pred_probs2)
print(roc_obj2)

plot(roc_obj2, main = "ROC Curve", print.auc = T,
    legacy.axes = TRUE, lwd = 2)
```

Appendix

Part 1:

```
### Q1.6

#** Generate gains table **

biz$pred_prob2 <- pred_probs2 <- predict(glm.model2, type =
"response")

str(biz)

install.packages("gains")
library(gains)
gains_table <- gains(actual = biz$is_open, predicted =
biz$pred_prob2,
                     groups = 10)
print(gains_table)
```

Appendix

Part 2:

```
### Q2.1

set.seed(123)
train_indices <- sample(1:nrow(biz), size = 0.7 * nrow(biz))

train_data <- biz[train_indices, ]
test_data <- biz[-train_indices, ]

# Model 1
glm.model1=glm(is_open ~ elite_cnt + price_level, data=biz,
family=binomial)
# Model 2
glm.model2= glm(is_open ~ elite_cnt + price_level * biz.stars +
repeated_cnt, biz, family =
                binomial)
```

```
# Predictions
train_pred1 <- predict(glm.model1, train_data, type = "response")
test_pred1 <- predict(glm.model1, test_data, type = "response")

train_pred2 <- predict(glm.model2, train_data, type = "response")
test_pred2 <- predict(glm.model2, test_data, type = "response")

library(pROC)
# Compare AUC
model1_auc <- c(
  auc(roc(train_data$is_open, train_pred1)),
  auc(roc(test_data$is_open, test_pred1)))

names(model1_auc) <- c("Train Set", "Test Set")
print(model1_auc)

model2_auc <- c(
  auc(roc(train_data$is_open, train_pred2)),
  auc(roc(test_data$is_open, test_pred2)))

names(model2_auc) <- c("Train Set", "Test Set")
print(model2_auc)
```

Appendix

Part 2:

```
### Q2.2

###****************** 10-fold cross-validation ************************
glm.model3= glm(is_open ~ poly(elite_cnt, 2, raw=T) +
price_level*biz.stars*repeated_cnt + city,
          biz, family=binomial)

summary(glm.model3)

# Set seed for reproducibility
set.seed(123)

# Define number of folds
k <- 10
folds <- cut(seq(1, nrow(biz)), breaks = k, labels = FALSE)
table(folds)

biz$is_open_f <- factor(ifelse(biz$is_open==1, "open", "closed"),
               levels=c("open","closed"))
str(biz)


# Load required library
library(pROC)
```

```
###********Model 1**********
# Initialize vector to store AUCs
auc_values1_test <- numeric(k)
auc_values1_train <- numeric(k)

# Perform k-fold CV
for(i in 1:k) {
  # Split into train and test sets
  test_indices <- which(folds == i)
  train_data <- biz[-test_indices, ]
  test_data <- biz[test_indices, ]

  model <- glm(is_open_f ~ elite_cnt + price_level,
         data = train_data, family = binomial)

  # Predict probabilities on the test set
  pred_probs1_train <- predict(model, newdata = train_data, type = "response")
  pred_probs1_test <- predict(model, newdata = test_data, type = "response")

  # Compute AUC
  roc_obj1_train <- roc(train_data$is_open_f, pred_probs1_train)
  roc_obj1_test <- roc(test_data$is_open_f, pred_probs1_test)
  auc_values1_train[i] <- auc(roc_obj1_train)
  auc_values1_test[i] <- auc(roc_obj1_test)
}

# Combine results into a data frame
cv_results1 <- data.frame(
  Fold = 1:k,
  AUC_Train = auc_values1_train,
  AUC_Test = auc_values1_test
)
cv_results1

# Mean AUC for training data
mean_auc_train1 <- mean(cv_results1$AUC_Train)

# Mean AUC for test data
mean_auc_test1 <- mean(cv_results1$AUC_Test)

# Print the results
mean_auc_train1 #0.6464
mean_auc_test1 #0.6373
```

```
###********Model 2**********
# Initialize vector to store AUCs
auc_values2_test <- numeric(k)
auc_values2_train <- numeric(k)

# Perform k-fold CV
for(i in 1:k) {
  # Split into train and test sets
  test_indices <- which(folds == i)
  train_data <- biz[-test_indices, ]
  test_data <- biz[test_indices, ]

  model <- glm(is_open ~ elite_cnt + price_level * biz.stars + repeated_cnt, biz, family =
           binomial)

  # Predict probabilities on the test set
  pred_probs2_train <- predict(model, newdata = train_data, type = "response")
  pred_probs2_test <- predict(model, newdata = test_data, type = "response")

  # Compute AUC
  roc_obj2_train <- roc(train_data$is_open_f, pred_probs2_train)
  roc_obj2_test <- roc(test_data$is_open_f, pred_probs2_test)
  auc_values2_train[i] <- auc(roc_obj2_train)
  auc_values2_test[i] <- auc(roc_obj2_test)
}

# Combine results into a data frame
cv_results2 <- data.frame(
  Fold = 1:k,
  AUC_Train = auc_values2_train,
  AUC_Test = auc_values2_test
)
cv_results2

# Mean AUC for training data
mean_auc_train2 <- mean(cv_results2$AUC_Train)

# Mean AUC for test data
mean_auc_test2 <- mean(cv_results2$AUC_Test)

# Print the results
mean_auc_train2 #0.7035
mean_auc_test2 #0.6962

# Train AUC ≈ Test AUC Stable performance. Trust the model.
```

```
v###********Model 3**********
# Initialize vector to store AUCs
auc_values3_test <- numeric(k)
auc_values3_train <- numeric(k)

# Perform k-fold CV
for(i in 1:k) {
  # Split into train and test sets
  test_indices <- which(folds == i)
  train_data <- biz[-test_indices, ]
  test_data <- biz[test_indices, ]

  model <- glm(is_open ~ poly(elite_cnt, 2, raw=T) + price_level*biz.stars*repeated_cnt +
city,
         biz, family=binomial)

  # Predict probabilities on the test set
  pred_probs3_train <- predict(model, newdata = train_data, type = "response")
  pred_probs3_test <- predict(model, newdata = test_data, type = "response")

  # Compute AUC
  roc_obj3_train <- roc(train_data$is_open_f, pred_probs3_train)
  roc_obj3_test <- roc(test_data$is_open_f, pred_probs3_test)
  auc_values3_train[i] <- auc(roc_obj3_train)
  auc_values3_test[i] <- auc(roc_obj3_test)
}

# Combine results into a data frame
cv_results3 <- data.frame(
  Fold = 1:k,
  AUC_Train = auc_values3_train,
  AUC_Test = auc_values3_test
)
cv_results3

# Mean AUC for training data
mean_auc_train3 <- mean(cv_results3$AUC_Train)

# Mean AUC for test data
mean_auc_test3 <- mean(cv_results3$AUC_Test)

# Print the results
mean_auc_train3 #0.7217
mean_auc_test3 #0.7122

# Train AUC ≈ Test AUC Stable performance. Trust the model.
```

Q2.2.2

```
# baseline is Model1's Test AUC
baseline    <- results$Test_AUC[results$Model=="Model1"]

# compute relative improvements
improv2     <- (results$Test_AUC[results$Model=="Model2"] - baseline) /
baseline
improv3     <- (results$Test_AUC[results$Model=="Model3"] - baseline) /
baseline

# pick best of Model2/3
if (improv2 > improv3) {
  best_model  <- "Model2"; best_improv <- improv2
} else {
  best_model  <- "Model3"; best_improv <- improv3
}

# decision logic
if (best_improv < 0.05) {
  cat("All more complex models improve < 5% → stick with Model1
(simplest).\n")
} else if (best_improv > 0.15) {
  cat(sprintf("%s is substantially better (%.1f%% ↑) → use %s.\n",
          best_model, best_improv*100, best_model))
} else {
  cat(sprintf("%s yields a moderate gain of %.1f%% (5–15%%) → not worth the
extra complexity; stick with Model1.\n",
          best_model, best_improv*100))
}
```