Code

Ex2 App Marketing Structure

Hide

```
library(data.table)
library(ggplot2)

# Clear All Variables & Clear the Screen
rm(list=ls())
cat("\014")
```

Hide

```
# Read in the Data
df = read.csv("Ex2_Data_R.csv")

# Explore the data
#str(df)
#summary(df)
```

0.1 Prepare Data

Hide

```
# Convert unit sales to log unit sales
df$logUNITS1 <- log(df$UNITS1)
df$logUNITS2 <- log(df$UNITS2)
df$logUNITS3 <- log(df$UNITS3)
```

1 Using a linear regressions framework with unit sales (and/or log unit sales) as the dependent variable, investigate the effect of regular price, feature and rating on sales for each of the three apps. How good are these models?

1.1 Simple Regression Model for App1 (only consider own price, feature and rating)

Hide

```
# Linear model
lm.linear.smp.1 = lm(UNITS1~REGPR1+FEAT1+RATING1, df)
# Log model
lm.log.smp.1 = lm(logUNITS1~REGPR1+FEAT1+RATING1, df)

# Calculate price elasticity for the linear model
price.coeff.linear.smp.1 = unname(lm.linear.smp.1$coefficients[2], force = FALSE)
elas.linear.smp.1        = price.coeff.linear.smp.1*mean(df$REGPR1)/mean(df$UNITS1)

# Calculate price elasticity for the log model
price.coeff.log.smp.1 = unname(lm.log.smp.1$coefficients[2], force = FALSE)
elas.log.smp.1        = price.coeff.log.smp.1*mean(df$REGPR1)

summary(lm.linear.smp.1)
```

```

Call:
lm(formula = UNITS1 ~ REGPR1 + FEAT1 + RATING1, data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-105.353  -58.484   -5.979   52.021  196.826 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   390.23     303.91   1.284 0.201857    
REGPR1       -537.37      77.52  -6.932 3.06e-10 ***
FEAT1         483.55      19.43  24.887  < 2e-16 ***
RATING1       233.10      64.03   3.640 0.000418 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 65.64 on 109 degrees of freedom
Multiple R-squared:  0.8863,    Adjusted R-squared:  0.8832 
F-statistic: 283.3 on 3 and 109 DF,  p-value: < 2.2e-16
```

Hide

```
summary(lm.log.smp.1)
```

```

Call:
lm(formula = logUNITS1 ~ REGPR1 + FEAT1 + RATING1, data = df)

Residuals:
      Min        1Q    Median        3Q       Max 
-0.129306 -0.075000 -0.003703  0.060155  0.243313 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  6.08183    0.38637  15.741  < 2e-16 ***
REGPR1      -0.70710    0.09855  -7.175 9.23e-11 ***
FEAT1        0.47511    0.02470  19.234  < 2e-16 ***
RATING1      0.32061    0.08140   3.939 0.000145 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08345 on 109 degrees of freedom
Multiple R-squared:  0.8395,    Adjusted R-squared:  0.835 
F-statistic:   190 on 3 and 109 DF,  p-value: < 2.2e-16
```

Comment: Both simple models fit quite well despite not including competitive effects. ALl three variables are significant.

1.2 Simple Regression Model for App2 (only consider own price, feature and rating)

Hide

```
# Linear model
lm.linear.smp.2 = lm(UNITS2~REGPR2+FEAT2+RATING2, df)
# Log model
lm.log.smp.2 = lm(logUNITS2~REGPR2+FEAT2+RATING2, df)

# Calculate price elasticity for the linear model
price.coeff.linear.smp.2 = unname(lm.linear.smp.2$coefficients[2], force = FALSE)
elas.linear.smp.2        = price.coeff.linear.smp.2*mean(df$REGPR2)/mean(df$UNITS2)

# Calculate price elasticity for the log model
price.coeff.log.smp.2 = unname(lm.log.smp.2$coefficients[2], force = FALSE)
elas.log.smp.2        = price.coeff.log.smp.2*mean(df$REGPR2)

summary(lm.linear.smp.2)
```

```

Call:
lm(formula = UNITS2 ~ REGPR2 + FEAT2 + RATING2, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-225.56  -62.36   -0.63   60.88  215.14 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  2429.13     498.76   4.870 3.80e-06 ***
REGPR2       -723.31     116.68  -6.199 1.04e-08 ***
FEAT2         406.36      35.49  11.451  < 2e-16 ***
RATING2      -395.37     130.34  -3.033  0.00302 ** 
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 93.97 on 109 degrees of freedom
Multiple R-squared:  0.6696,	Adjusted R-squared:  0.6605 
F-statistic: 73.64 on 3 and 109 DF,  p-value: < 2.2e-16
```

Hide

```
summary(lm.log.smp.2)
```

```

Call:
lm(formula = logUNITS2 ~ REGPR2 + FEAT2 + RATING2, data = df)

Residuals:
     Min       1Q   Median       3Q      Max 
-2.05656 -0.16370  0.06079  0.27914  0.83912 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  12.3389     2.4657   5.004 2.17e-06 ***
REGPR2       -2.8905     0.5768  -5.011 2.11e-06 ***
FEAT2         0.9329     0.1754   5.318 5.64e-07 ***
RATING2      -1.0956     0.6444  -1.700   0.0919 .  
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.4646 on 109 degrees of freedom
Multiple R-squared:  0.3963,	Adjusted R-squared:  0.3796 
F-statistic: 23.85 on 3 and 109 DF,  p-value: 6.095e-12
```

Comment: The simple linear model fits better with a R-square of 0.67. All three variables are significant

1.3 Simple Regression Model for App3 (only consider own price, feature and rating)

Hide

```
# Linear model
lm.linear.smp.3 = lm(UNITS3~REGPR3+FEAT3+RATING3, df)
# Log model
lm.log.smp.3 = lm(logUNITS3~REGPR3+FEAT3+RATING3, df)

# Calculate price elasticity for the linear model
price.coeff.linear.smp.3 = unname(lm.linear.smp.3$coefficients[2], force = FALSE)
elas.linear.smp.3        = price.coeff.linear.smp.3*mean(df$REGPR3)/mean(df$UNITS3)

# Calculate price elasticity for the log model
price.coeff.log.smp.3 = unname(lm.log.smp.3$coefficients[2], force = FALSE)
elas.log.smp.3        = price.coeff.log.smp.3*mean(df$REGPR3)

summary(lm.linear.smp.3)
```

```
Call:
lm(formula = UNITS3 ~ REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-1457.8  -143.5   -35.2    59.3  5578.2 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   -735.5     1565.2  -0.470    0.639    
REGPR3        -970.3      702.0  -1.382    0.170    
FEAT3         1914.1      219.2   8.732 3.29e-14 ***
RATING3        483.8      356.5   1.357    0.178    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 650.7 on 109 degrees of freedom
Multiple R-squared:  0.4764,    Adjusted R-squared:  0.462 
F-statistic: 33.05 on 3 and 109 DF,  p-value: 2.843e-15
```

Hide

```
summary(lm.log.smp.3)
```

```
Call:
lm(formula = logUNITS3 ~ REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
    Min      1Q  Median      3Q     Max 
-3.5640 -0.7373  0.1142  0.7525  2.3662 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  -1.7395     2.6620  -0.653 0.514853    
REGPR3       -2.7405     1.1939  -2.295 0.023619 *  
FEAT3         2.6005     0.3728   6.975 2.48e-10 ***
RATING3       2.3246     0.6064   3.834 0.000212 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.107 on 109 degrees of freedom
Multiple R-squared:  0.4475,    Adjusted R-squared:  0.4323 
F-statistic: 29.43 on 3 and 109 DF,  p-value: 5.122e-14
```

Comment: Neither model fits particularly well as the R-squares are relatively small. The simple linear model only includes the feature varibale and leaves out the price. The semi-log one accounts for more variables but has a lower R-square.

2 An important source of variation in sales often comes from competitive marketing activity. Investigate the impact on sales of each app from the changes in the marketing activity of competing apps. What, if any, competitive terms would you want to include in your final models?

2.1 Full Regression Model for App1 (consider both own and competitors’ price, feature and rating)

Hide

```
# Linear model for app1
lm.linear.1 = lm(UNITS1~REGPR1+FEAT1+RATING1+REGPR2+FEAT2+RATING2+REGPR3+FEAT3+RATING3, df)
# Log model for app1
lm.log.1 = lm(logUNITS1~REGPR1+FEAT1+RATING1+REGPR2+FEAT2+RATING2+REGPR3+FEAT3+RATING3, df)

# Calculate price elasticity for linear model
price.coeff.linear.1 = unname(lm.linear.1$coefficients[2], force = FALSE)
elas.linear.1        = price.coeff.linear.1*mean(df$REGPR1)/mean(df$UNITS1)

# Calculate price elasticity for log model
price.coeff.log.1 = unname(lm.log.1$coefficients[2], force = FALSE)
elas.log.1        = price.coeff.log.1*mean(df$REGPR1)

summary(lm.linear.1)
```

```

Call:
lm(formula = UNITS1 ~ REGPR1 + FEAT1 + RATING1 + REGPR2 + FEAT2 +
    RATING2 + REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-104.530  -52.349   -4.502   45.181  195.209

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 1059.5182   948.6804   1.117    0.267
REGPR1      -614.1419   111.5209  -5.507  2.7e-07 ***
FEAT1        479.4691    20.9037  22.937  < 2e-16 ***
RATING1      187.8424   131.3334   1.430    0.156
REGPR2       171.9486   111.2072   1.546    0.125
FEAT2          0.1793    26.1158   0.007    0.995
RATING2      -82.7993   143.5679  -0.577    0.565
REGPR3       -96.9208    94.6579  -1.024    0.308
FEAT3        -35.2794    23.3590  -1.510    0.134
RATING3      -42.8413    71.1072  -0.602    0.548
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 66.13 on 103 degrees of freedom
Multiple R-squared:  0.891, Adjusted R-squared:  0.8815
F-statistic: 93.54 on 9 and 103 DF,  p-value: < 2.2e-16
```

Hide

```
summary(lm.log.1)
```

```

Call:
lm(formula = logUNITS1 ~ REGPR1 + FEAT1 + RATING1 + REGPR2 +
    FEAT2 + RATING2 + REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
      Min        1Q    Median        3Q       Max
-0.128163 -0.063482 -0.005767  0.060318  0.233271

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  6.6362753  1.1998782   5.531 2.43e-07 ***
REGPR1      -0.8118486  0.1410502  -5.756 8.95e-08 ***
FEAT1        0.4689827  0.0264387  17.738  < 2e-16 ***
RATING1      0.2894567  0.1661087   1.743   0.0844 .
REGPR2       0.2844113  0.1406533   2.022   0.0458 *
FEAT2       -0.0006932  0.0330309  -0.021   0.9833
RATING2     -0.0965949  0.1815828  -0.532   0.5959
REGPR3      -0.1385222  0.1197221  -1.157   0.2499
FEAT3       -0.0402567  0.0295442  -1.363   0.1760
RATING3     -0.0259559  0.0899354  -0.289   0.7735
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.08364 on 103 degrees of freedom
Multiple R-squared:  0.8476,    Adjusted R-squared:  0.8343
F-statistic: 63.65 on 9 and 103 DF,  p-value: < 2.2e-16
```

2.2 Full Regression Model for App2 (consider both own and competitors’ price, feature and rating)

Hide

```
# Linear model for app1
lm.linear.2 = lm(UNITS2~REGPR1+FEAT1+RATING1+REGPR2+FEAT2+RATING2+REGPR3+FEAT3+RATING3, df)
# Log model for app1
lm.log.2 = lm(logUNITS2~REGPR1+FEAT1+RATING1+REGPR2+FEAT2+RATING2+REGPR3+FEAT3+RATING3, df)

# Calculate price elasticity for linear model
price.coeff.linear.2 = unname(lm.linear.2$coefficients[5], force = FALSE)
elas.linear.2        = price.coeff.linear.2*mean(df$REGPR2)/mean(df$UNITS2)

# Calculate price elasticity for log model
price.coeff.log.2 = unname(lm.log.2$coefficients[5], force = FALSE)
elas.log.2        = price.coeff.log.2*mean(df$REGPR2)

summary(lm.linear.2)
```

```

Call:
lm(formula = UNITS2 ~ REGPR1 + FEAT1 + RATING1 + REGPR2 + FEAT2 +
    RATING2 + REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-167.302  -52.337    1.311   52.771  177.715

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1394.562   1136.592   1.227   0.2226
REGPR1        558.690    133.611   4.181 6.10e-05 ***
FEAT1         -64.168     25.044  -2.562   0.0118 *
RATING1      -162.651    157.347  -1.034   0.3037
REGPR2      -1016.092    133.235  -7.626 1.25e-11 ***
FEAT2         363.327     31.289  11.612  < 2e-16 ***
RATING2         7.055    172.005   0.041   0.9674
REGPR3        107.956    113.407   0.952   0.3434
FEAT3           3.725     27.986   0.133   0.8944
RATING3       -46.913     85.192  -0.551   0.5830
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 79.23 on 103 degrees of freedom
Multiple R-squared:  0.7781,    Adjusted R-squared:  0.7587
F-statistic: 40.12 on 9 and 103 DF,  p-value: < 2.2e-16
```

Hide

```
summary(lm.log.2)
```

```

Call:
lm(formula = logUNITS2 ~ REGPR1 + FEAT1 + RATING1 + REGPR2 +
    FEAT2 + RATING2 + REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-1.23353 -0.19071  0.03763  0.18847  0.88096

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   4.0289     5.5944   0.720  0.47305
REGPR1        2.1871     0.6576   3.326  0.00122 **
FEAT1        -0.5338     0.1233  -4.331 3.46e-05 ***
RATING1       0.8106     0.7745   1.047  0.29770
REGPR2       -4.0459     0.6558  -6.170 1.36e-08 ***
FEAT2         0.7508     0.1540   4.875 3.96e-06 ***
RATING2      -0.3965     0.8466  -0.468  0.64056
REGPR3        0.2189     0.5582   0.392  0.69576
FEAT3        -0.1001     0.1377  -0.727  0.46903
RATING3       0.2423     0.4193   0.578  0.56460
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.39 on 103 degrees of freedom
Multiple R-squared:  0.598, Adjusted R-squared:  0.5629
F-statistic: 17.02 on 9 and 103 DF,  p-value: < 2.2e-16
```

2.3 Full Regression Model for App3 (consider both own and competitors’ price, feature and rating)

Hide

```
# Linear model for app1
lm.linear.3 = lm(UNITS3~REGPR1+FEAT1+RATING1+REGPR2+FEAT2+RATING2+REGPR3+FEAT3+RATING3, df)
# Log model for app1
lm.log.3 = lm(logUNITS3~REGPR1+FEAT1+RATING1+REGPR2+FEAT2+RATING2+REGPR3+FEAT3+RATING3, df)

# Calculate price elasticity for linear model
price.coeff.linear.3 = unname(lm.linear.3$coefficients[8], force = FALSE)
elas.linear.3        = price.coeff.linear.3*mean(df$REGPR3)/mean(df$UNITS3)

# Calculate price elasticity for log model
price.coeff.log.3 = unname(lm.log.3$coefficients[8], force = FALSE)
elas.log.3        = price.coeff.log.3*mean(df$REGPR3)

summary(lm.linear.3)
```

```
Call:
lm(formula = UNITS3 ~ REGPR1 + FEAT1 + RATING1 + REGPR2 + FEAT2 +
    RATING2 + REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-1482.8  -142.9   -26.6    67.1  5553.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5377.77    9481.06  -0.567    0.572
REGPR1       1453.09    1114.53   1.304    0.195
FEAT1          44.29     208.91   0.212    0.833
RATING1       160.64    1312.54   0.122    0.903
REGPR2         43.61    1111.40   0.039    0.969
FEAT2          83.65     261.00   0.321    0.749
RATING2       559.75    1434.81   0.390    0.697
REGPR3      -1430.33     946.01  -1.512    0.134
FEAT3        1883.66     233.45   8.069 1.37e-12 ***
RATING3       660.28     710.64   0.929    0.355
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 660.9 on 103 degrees of freedom
Multiple R-squared:  0.4896,    Adjusted R-squared:  0.445
F-statistic: 10.98 on 9 and 103 DF,  p-value: 7.811e-12
```

Hide

```
summary(lm.log.3)
```

```
Call:
lm(formula = logUNITS3 ~ REGPR1 + FEAT1 + RATING1 + REGPR2 +
    FEAT2 + RATING2 + REGPR3 + FEAT3 + RATING3, data = df)

Residuals:
     Min       1Q   Median       3Q      Max
-2.24273 -0.62420 -0.00151  0.68602  2.34948

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -7.9024    14.4117  -0.548 0.584651
REGPR1        6.1987     1.6942   3.659 0.000401 ***
FEAT1        -0.1235     0.3176  -0.389 0.698189
RATING1       1.2626     1.9951   0.633 0.528226
REGPR2       -0.6064     1.6894  -0.359 0.720378
FEAT2         0.3277     0.3967   0.826 0.410736
RATING2      -1.1690     2.1810  -0.536 0.593104
REGPR3       -4.7558     1.4380  -3.307 0.001298 **
FEAT3         2.4898     0.3549   7.016 2.48e-10 ***
RATING3       2.5250     1.0802   2.338 0.021345 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.005 on 103 degrees of freedom
Multiple R-squared:  0.5699,    Adjusted R-squared:  0.5323
F-statistic: 15.16 on 9 and 103 DF,  p-value: 1.944e-15
```

3 Propose the “best” regression model for each app, taking into account own-effects and competitive-effects.

Comment on the quality of these final models.

3.1 App1

Hide

```
# Compare models for App1
#summary(lm.linear.smp.1)
#summary(lm.log.smp.1)
#summary(lm.linear.1)
#summary(lm.log.1) # best model: High R-square, taken into account of competitor's price
```

own models, own&comp models
 We propose the ‘semi-log own&comp’(lm.log.1) as the best model. Though the ‘linear own&comp’ has the highest R-square, it doesn’t take competitor’s prices into account. The semi-log has not only app1’s price and feature significant, but also app2’s price. The price elasticity for this model is -0.9, which implies app1 is not very suscepitble to price change.

3.2 App2

Hide

```
# Compare models for App2
#summary(lm.linear.smp.2)
#summary(lm.log.smp.2)
#summary(lm.linear.2)  # best model: Highest R-square, incorporate the cross-price effect by app1
#summary(lm.log.2)
```

own models, own&comp models
 We propose the ‘linear own&comp’(lm.linear.1) as the best model. Not only it has the highest R-Square, it has taken App1’s price and feature into account. The price elasticity for this model is -3.7, implies app2’s sale is relatively susceptible to price change.

3.3 App3

Hide

```

# Compare models for App3
#summary(lm.linear.smp.3)
#summary(lm.log.smp.3)
#summary(lm.linear.3)
#summary(lm.log.3)    # best model: Highest R-square, incorporate the cross-price effect by app1
```

own models, own&comp models
 We propose the ‘log own&comp’(lm.log.3) as the best model. Not only it has the highest R-Square, it has taken App1’s price into account. The price elasticity for this model is -4.6, implies app3’s sale is very susceptible to price change.

4 What do these best models tell you about market structure and inter-app competition in this game category, e.g., create a clout/vulnerability map (as we did in class)?

4.1 Construct a price coefficient table

Hide

```
# Build a table of price coefficients with cross-effects
# The selected model is named by: model_type.app.app_number
# Also, only significant price coefficients are selected
semi_log.app.1 <- c(lm.log.1$coefficients[2], lm.log.1$coefficients[5], NA)
linear.app.2 <- c(lm.linear.2$coefficients[2], lm.linear.2$coefficients[5], NA)
semi_log.app.3 <- c(lm.log.3$coefficients[2], NA, lm.log.3$coefficients[8])

price_coeff_table <- data.frame(semi_log.app.1,linear.app.2,semi_log.app.3)
row.names(price_coeff_table) <- c("price.coeff.app.1", "price.coeff.app.2", "price.coeff.app.3")
price_coeff_table <- data.frame(t(price_coeff_table))
price_coeff_table
```

	price.coeff.app.1 <dbl>	price.coeff.app.2 <dbl>	price.coeff.app.3 <dbl>
semi_log.app.1	-0.8118486	0.2844113	NA
linear.app.2	558.6898788	-1016.0917892	NA

	price.coeff.app.1	price.coeff.app.2	price.coeff.app.3
	<dbl>	<dbl>	<dbl>
semi_log.app.3	6.1986618	NA	-4.755815

3 rows

4.2 Calculate the cross-price elasticity

Hide

```
# Convert price coefficients to elasticity
price_elas_table <- price_coeff_table
colnames(price_elas_table) <-c("price.elas.app.1","price.elas.app.2","price.elas.app.3")

price_elas_table$price.elas.app.1[1] <- price_elas_table$price.elas.app.1[1]*(mean(df$REGPR1))
price_elas_table$price.elas.app.1[2] <- price_elas_table$price.elas.app.1[2]*(mean(df$REGPR1)/mean(df$UNITS2))
price_elas_table$price.elas.app.1[3] <- price_elas_table$price.elas.app.1[3]*mean(df$REGPR1)
price_elas_table$price.elas.app.2[1] <- price_elas_table$price.elas.app.2[1]*(mean(df$REGPR2))
price_elas_table$price.elas.app.2[2] <- price_elas_table$price.elas.app.2[2]*(mean(df$REGPR2)/mean(df$UNITS2))
price_elas_table$price.elas.app.3[3] <- price_elas_table$price.elas.app.3[3]*mean(df$REGPR3)
price_elas_table
```

	price.elas.app.1	price.elas.app.2	price.elas.app.3
	<dbl>	<dbl>	<dbl>
semi_log.app.1	-0.9068277	0.2861228	NA
linear.app.2	2.2317890	-3.6557051	NA
semi_log.app.3	6.9238504	NA	-4.578629

3 rows

4.3 Build the clout and vulnerability table

Hide

```
# Build a clout and vulnerability table by adding the clout and vulnerability up for each app
clout.app.1 <- price_elas_table$price.elas.app.1[2]+price_elas_table$price.elas.app.1[3]
clout.app.2 <- price_elas_table$price.elas.app.2[1]
clout.app.3 <- 0

vulnerability.app.1 <- price_elas_table$price.elas.app.2[1]
vulnerability.app.2 <- price_elas_table$price.elas.app.1[2]
vulnerability.app.3 <- price_elas_table$price.elas.app.1[3]

clout_and_vulnerability <- data.frame(c("app1","app2", "app3"),
                                      c(clout.app.1,clout.app.2,clout.app.3),
                                      c(vulnerability.app.1,vulnerability.app.2,vulnerability.app.3))
colnames(clout_and_vulnerability) <- c("app", "clout", "vulnerability")
clout_and_vulnerability
```

app	clout	vulnerability
<chr>	<dbl>	<dbl>
app1	9.1556393	0.2861228
app2	0.2861228	2.2317890
app3	0.0000000	6.9238504

3 rows

comment on clout and vulnerability table interpretation of clouts: if app1 drops its price by 1%, the sales of all competing apps will drop by 9.16% if app2 drops its price by 1%, the sales of all competing apps will drop by 0.29%

interpretation of vulnerability: if the price of all competing apps drop 1%,the sales of app1 will drop by 0.29% if the price of all competing apps drop 1%,the sales of app2 will drop by 2.23% if the price of all competing apps drop 1%,the sales of app3 will drop by 6.92%

4.4 Visualize clout and vulnerability

Hide

```
# add market share column
# market share is calculated by the percentage of average unit size
avg.unit.1 <- mean(df$UNITS1)
avg.unit.2 <- mean(df$UNITS2)
avg.unit.3 <- mean(df$UNITS3)
total.unit <- avg.unit.1+avg.unit.2+avg.unit.3
clout_and_vulnerability$share <- c(100*avg.unit.1/total.unit, 100*avg.unit.2/total.unit, 100*avg.unit.3/total.uni
t)

# plot the clout & vulnerability chart
ggplot(data=clout_and_vulnerability, mapping=aes(x=vulnerability, y=clout, label=app)) + geom_point(aes(size=shar
e), color="coral1") + geom_text(nudge_x=0.4) + ggtitle("Clout and Vulnerability")
```