423 project

Chris Chen

1/20/2022

Preliminaries

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.1 ──
```

```
## ✓ ggplot2 3.3.5     ✓ purrr   0.3.4
## ✓ tibble  3.1.6     ✓ dplyr   1.0.7
## ✓ tidyr   1.1.4     ✓ stringr 1.4.0
## ✓ readr   2.1.0     ✓ forcats 0.5.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(expm)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
##
## Attaching package: 'expm'
```

```
## The following object is masked from 'package:Matrix':
##
##     expm
```

```
library(ggplot2)
library(lmvar)
library(leaps)
```

Dataset

```
perfume = read.csv("noon_perfumes_dataset.csv")
sum(is.na(perfume))
```

```
## [1] 0
```

```
head(perfume)
```

	X <int>	brand <chr>	name <chr>	old_price <dbl>	new_price <dbl>	ml <int>	concentration <chr>	departmer <chr>
1	0	PACO RABANNE	1 Million Lucky	395	244.55	100	EDT	Men
2	1	Roberto Cavalli	Paradiso Assoluto	415	107.95	50	EDP	Women
3	2	S.T.Dupont	Royal Amber	265	186.90	100	EDP	Unisex
4	3	GUESS	Seductive Blue	290	103.20	100	EDT	Men
5	4	Roberto Cavalli	Uomo	260	94.95	50	EDP	Women
6	5	Roberto Cavalli	cavalli	260	94.95	50	EDP	Women

6 rows | 1-10 of 16 columns

no empty value. good.

```
perfume = perfume %>%
  mutate(scent = ifelse(scents == "Arabian", "Oriental", scents))
p1 = subset(perfume, scent != "Vanilla" & scent != "Aromatic" & scent != "Musk" & scent
 != "Jasmine" & scent != "Floral and Oriental" & scent != "Rose, Floral" & scent != "San
dalwood" & scent != "Woody, Sweet" & scent != "Aromatic,Citrus" & scent != "Clean" & sce
nt != "Oriental, Floral" & scent != "Sweet Aromatic" & scent != "Woody And Spicy" & scen
t != "Woody, Musky")
```

```
p2 = p1 %>%
  mutate(conc = ifelse(concentration == "PDT", "EDT", concentration))
p2 = subset(p2, select = -c(concentration))
```

```
p3 = p2 %>%
  mutate(brands1 = ifelse(brand == "ST Dupont", "S.T.Dupont", brand)) %>%
  mutate(brands2 = ifelse(brands1 == "armani", "GIORGIO ARMANI", brands1)) %>%
  mutate(brands3 = ifelse(brands2 == "Genie Collection", "Genie", brands2)) %>%
  mutate(brands4 = ifelse(brands3 == "LANVIN PARIS", "LANVIN", brands3)) %>%
  mutate(brands5 = ifelse(brands4 == "Mont Blanc", "MONTBLANC", brands4)) %>%
  mutate(brands6 = ifelse(brands5 == "marbert man", "Marbert", brands5)) %>%
  mutate(brands = ifelse(brands6 == "YSL" | brands6 == "YVES", "Yves Saint Laurent", bra
nds6))
p3 = subset(p3, select = -c(brand, brands1, brands2, brands3, brands4, brands5, brands
6))
```

```
p4 = subset(p3, seller_rating <= 5.0)
p5 = p4 %>%
  mutate(num_sel_ratings =
           ifelse(grepl("K", num_seller_ratings),
                  as.numeric(substring(num_seller_ratings, 1, nchar(num_seller_ratings)
 - 1)) * 1000,
                  as.numeric(num_seller_ratings)))
```

```
## Warning in ifelse(grepl("K", num_seller_ratings),
## as.numeric(substring(num_seller_ratings, : NAs introduced by coercion
```

```
p5 = subset(p5, select = -c(num_seller_ratings))
```

```
# clean seller column
seller = as.vector(p5$seller)
seller = tolower(seller)
index_golden = which(grepl("golden", seller))
seller[index_golden] = "golden perfumes"
index_lolita = which(grepl("lolita", seller))
seller[index_lolita] = "lolita shop"
index_noon = which(grepl("noon", seller))
seller[index_noon] = "noon"
index_swiss = which(grepl("swiss", seller))
seller[index_swiss] = "swiss arabian perfumes"
index_pa = which(grepl("perfumes--addresses", seller))
seller[index_pa] = "perfumes"
index_ps = which(grepl("perfumes-shop", seller))
seller[index_ps] = "perfumes"

p6 = p5
p6$seller = seller
sb = c(48, 435, 651)
bf = c(109, 121, 470, 565, 576)
p6 = p6 %>%
  mutate(seller1 = ifelse(is.element(X, sb), "show biz", seller)) %>%
  mutate(sellers = ifelse(is.element(X, bf), "beauty fortune", seller))
p6 = subset(p6, select = -c(seller1, seller))
```

```
base_note = as.vector(p6$base_note)
base_note = tolower(base_note)
base_note = str_replace_all(base_note, " and ", ",")
base_note = str_replace_all(base_note, " ", "")
base_note = str_replace_all(base_note, "vanille", "vanilla")
base_note = str_replace_all(base_note, "woodsynotes", "wood")
base_note = str_replace_all(base_note, "orrisroot", "orris")
base_note = str_replace_all(base_note, "woodsynote", "wood")
base_note = str_replace_all(base_note, "woodynotes", "wood")
base_note = str_replace_all(base_note, "woody", "wood")
base_note = str_replace_all(base_note, "cedarwood", "cedar")
base_note = str_replace_all(base_note, "virginiacedar", "cedar")
base_note = str_replace_all(base_note, "whitemusk", "musk")
base_note = str_replace_all(base_note, "tonkabeans", "tonka")
base_note = str_replace_all(base_note, "tonkabean", "tonka")
base_note = str_replace_all(base_note, "amberwood", "amber")
base_note = str_replace_all(base_note, "sandalwood", "sandal")
base_note = str_replace_all(base_note, "cashmerewood", "cashmere")
base_note = str_replace_all(base_note, "guaiacwood", "guaiac")
base_note = str_replace_all(base_note, "ambergris", "AMBERGRIS")
base_note = str_replace_all(base_note, "mustyoud", "oud")
base_note = str_replace_all(base_note, "naturaloudoil", "oud")
base_note = str_replace_all(base_note, "agarwood\\(oud\\)", "oud")
base_note = str_replace_all(base_note, "agarwood", "oud")
base_note = str_replace_all(base_note, "oudh", "oud")
p6$base_note = base_note
```

```
mid_note = as.vector(p6$middle_note)
mid_note = tolower(mid_note)
mid_note = str_replace_all(mid_note, " and ", ",")
mid_note = str_replace_all(mid_note, " ", "")
mid_note = str_replace_all(mid_note, "lily-of-the-valley", "lily")
mid_note = str_replace_all(mid_note, "orrisroot", "orris")
mid_note = str_replace_all(mid_note, "lilyofthevalley", "lily")
mid_note = str_replace_all(mid_note, "bulgarianrose", "rose")
mid_note = str_replace_all(mid_note, "africanorangeflower", "orangeblossom")
mid_note = str_replace_all(mid_note, "neroli", "orangeblossom")
mid_note = str_replace_all(mid_note, "jasminesambac", "jasmine")
mid_note = str_replace_all(mid_note, "wildjasmine", "jasmine")
mid_note = str_replace_all(mid_note, "wildjasmine", "jasmine")
mid_note = str_replace_all(mid_note, "blackpepper", "pepper")
mid_note = str_replace_all(mid_note, "pinkpepper", "pepper")
mid_note = str_replace_all(mid_note, "vanille", "vanilla")
mid_note = str_replace_all(mid_note, "tuberose", "TUBEROSE")
mid_note = str_replace_all(mid_note, "orrisroot", "ORRISROOT")
mid_note = str_replace_all(mid_note, "honeysuckle", "HONEYSUCKLE")
mid_note = str_replace_all(mid_note, "rosemary", "ROSEMARY")
mid_note = str_replace_all(mid_note, "violetleaf", "VIOLETLEAF")
mid_note = str_replace_all(mid_note, "clarysage", "CLARYSAGE")
mid_note = str_replace_all(mid_note, "oudh", "oud")
mid_note = str_replace_all(mid_note, "burningoud", "oud")
mid_note = str_replace_all(mid_note, "agarwood\\(oud\\)", "oud")
mid_note = str_replace_all(mid_note, "agarwood", "oud")
mid_note = str_replace_all(mid_note, "oudwood", "oud")
p6$middle_note = mid_note
```

```
# clean ml column
vol = as.vector(p6$ml)
del_vol = as.data.frame(vol) %>%
  group_by(vol) %>%
  summarise(count = n()) %>%
  filter(count <= 5) %>%
  subset(select = vol)
del_vol = as.vector(del_vol$vol)

p7 = p6
index_del = which(p7$ml %in% del_vol)
p7 = p7[-index_del, ]

# add ordinal version of ml
vol = as.vector(p7$ml)
unique_vol = as.data.frame(vol) %>%
  group_by(vol) %>%
  summarise(count = n()) %>%
  subset(select = vol)
unique_vol = as.vector(unique_vol$vol)

order = vol
rank = 0
for (i in unique_vol) {
  rank = rank + 1
  index = which(vol == i)
  order[index] = rank
}
p7$ml_order = order
p7 = subset(p7, select = -c(ml))
```

```
perfume = subset(p7, select = -c(X, name, scents))
perfume = unique(perfume)

brand = as.vector(p7$brands)
brand = tolower(brand)
new_brands = as.data.frame(brand) %>%
  group_by(brand) %>%
  summarise(count = n()) %>%
  arrange(desc(count))
big_brands = new_brands[which(new_brands$count > 10), ]$brand
perfume = perfume %>%
  mutate(big_brand = ifelse(is.element(tolower(brands), big_brands), 1, 0))
perfume = subset(perfume, select = -c(brands))

perfume = perfume %>%
  mutate(is_noon = ifelse(tolower(sellers) == 'noon', 1, 0))
perfume = subset(perfume, select = -c(sellers))

get_notes = function(base, middle) {
  bnote = as.vector(unlist(strsplit(base, split = ",")))
  mnote = as.vector(unlist(strsplit(middle, split = ",")))
  return(union(bnote, mnote))
}

complexity = function(notes) {
  return(length(notes))
}

luxury = function(notes) {
  score = 0
  for (i in 1:length(notes)) {
    if (notes[i] == "musk" | notes[i] == "orris") { # 100-200
      score = score + 1
    } else if (notes[i] == "neroli" | notes[i] == "jasmine" | notes[i] == "sandal") { #
 200-400
      score = score + 2
    } else if (notes[i] == "rose" | notes[i] == "tuberose") { # 400-800
      score = score + 3
    } else if (notes[i] == "AMBERGRIS") { # 800-1200
      score = score + 4
    } else if (notes[i] == "oud") { # 1200-1600
      score = score + 5
    } else {
      score = score + 0
    }
  }
  return(score)
}
```

```
N = nrow(perfume)
complex = lux = rep(0, N)
for (i in 1:N) {
  complex[i] = complexity(get_notes(perfume[i, ]$base_note, perfume[i, ]$middle_note))
  lux[i] = luxury(get_notes(perfume[i, ]$base_note, perfume[i, ]$middle_note))
}
comp_score = lux_score = rep(0, N)
for (i in 1:N) {
  x = complex[i]
  comp_score[i] = sum(complex <= x) / N * 100
  y = lux[i]
  lux_score[i] = sum(lux <= y) / N * 100
}
nose_score = comp_score * lux_score / 100  # separate to two, include interaction
perfume = perfume %>%
  mutate(nose_rating = nose_score)
```

```
rse = function(model) {
  sqrt(sum(model$residuals ^ 2) / model$df.residual)
}

r2 = function(model) {
  summary(model)$adj.r.squared
}

mse = function(model) {
  mean(model$residuals ^ 2)
}

ge = function(model) {
  n = nobs(model)
  ge = 2 * (rse(model) ^ 2) * length(model$coefficients) / n
  return(ge)
}

Cp.lm = function(mdl.list) {
  n = nobs(mdl.list[[1]])
  DoFs = sapply(mdl.list, function(mdl) { sum(hatvalues(mdl)) })
  MSEs = sapply(mdl.list, function(mdl) { mean(residuals(mdl)^2) })
  biggest = which.max(DoFs)
  sigma2.hat = MSEs[[biggest]]*n/(n-DoFs[[biggest]])
  Cp = MSEs + 2*sigma2.hat*DoFs/n
  return(Cp)
}
```

```
lm.1 = lm(old_price ~ big_brand + nose_rating + item_rating + is_noon +
            conc + ml_order + num_sel_ratings +
            department + seller_rating + scent, data = perfume)
summary(lm.1)
```

```
##
## Call:
## lm(formula = old_price ~ big_brand + nose_rating + item_rating +
##     is_noon + conc + ml_order + num_sel_ratings + department +
##     seller_rating + scent, data = perfume)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -397.23 -146.26  -17.12  115.35 1927.79
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)      -5.236e+02  3.607e+02  -1.452   0.1470
## big_brand         7.519e+01  1.678e+01   4.482 8.53e-06 ***
## nose_rating      -7.140e-01  3.078e-01  -2.320   0.0206 *
## item_rating       7.597e+00  1.451e+01   0.523   0.6008
## is_noon           9.172e+01  9.847e+01   0.931   0.3519
## concEDP           2.241e+02  2.296e+02   0.976   0.3294
## concEDT           6.419e+01  2.290e+02   0.280   0.7793
## ml_order          1.569e+01  3.662e+00   4.286 2.05e-05 ***
## num_sel_ratings  -1.271e-03  1.017e-03  -1.250   0.2116
## departmentMen     2.894e+02  2.304e+02   1.256   0.2095
## departmentUnisex  1.829e+02  2.321e+02   0.788   0.4310
## departmentWomen   2.768e+02  2.292e+02   1.207   0.2276
## seller_rating     6.794e+01  4.147e+01   1.638   0.1018
## scentFloral      -8.053e+00  3.169e+01  -0.254   0.7995
## scentFresh       -9.492e+01  4.476e+01  -2.121   0.0342 *
## scentFruity      -5.218e+01  3.955e+01  -1.319   0.1874
## scentOriental    -6.088e+01  3.746e+01  -1.625   0.1046
## scentSpicy       -3.624e+01  3.492e+01  -1.038   0.2997
## scentWoody        1.735e+01  3.203e+01   0.542   0.5882
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 227.5 on 777 degrees of freedom
## Multiple R-squared:  0.1439, Adjusted R-squared:  0.1241
## F-statistic: 7.256 on 18 and 777 DF,  p-value: < 2.2e-16
```

```
# remove is_noon
lm.2 = lm(old_price ~ big_brand + nose_rating + item_rating +
            conc + ml_order + num_sel_ratings +
            department + seller_rating + scent, data = perfume)
# remove item_rating
lm.3 = lm(old_price ~ big_brand + nose_rating +
            conc + ml_order + num_sel_ratings +
            department + seller_rating + scent, data = perfume)
# remove num_sel_ratings
lm.4 = lm(old_price ~ big_brand + nose_rating +
            conc + ml_order +
            department + seller_rating + scent, data = perfume)
# remove seller_rating
lm.5 = lm(old_price ~ big_brand + nose_rating +
            conc + ml_order +
            department + scent, data = perfume)
# remove department
lm.6 = lm(old_price ~ big_brand + nose_rating +
            conc + ml_order +
            scent, data = perfume)
# remove concentration
lm.7 = lm(old_price ~ big_brand + nose_rating +
            ml_order + scent, data = perfume)
# remove scent
lm.8 = lm(old_price ~ big_brand + nose_rating + ml_order, data = perfume)
# RSE
rses = c(rse(lm.1), rse(lm.2), rse(lm.3), rse(lm.4), rse(lm.5), rse(lm.6), rse(lm.7), rs
e(lm.8)); rses
```

```
## [1] 227.4622 227.4428 227.3374 227.5173 227.5165 228.5496 237.9832 239.7241
```

```
# R^2
r2s = c(r2(lm.1), r2(lm.2), r2(lm.3), r2(lm.4), r2(lm.5), r2(lm.6), r2(lm.7), r2(lm.8));
r2s
```

```
## [1] 0.12407974 0.12422881 0.12504069 0.12365521 0.12366175 0.11568496 0.04117638
## [8] 0.02709730
```

```
# MSE
mses = c(mse(lm.1), mse(lm.2), mse(lm.3), mse(lm.4), mse(lm.5), mse(lm.6), mse(lm.7), ms
e(lm.8)); mses
```

```
## [1] 50504.07 50560.46 50578.52 50723.64 50788.29 51447.45 55924.49 57178.85
```

```
# generalization error
ges = c(ge(lm.1), ge(lm.2), ge(lm.3), ge(lm.4), ge(lm.5), ge(lm.6), ge(lm.7), ge(lm.8));
ges
```

```
## [1] 2469.9546 2339.5587 2207.5349 2080.9698 1950.8947 1574.9220 1423.0151
## [8]  577.5641
```

```
# Marlow's Cp
Cp.lm(list(lm.1, lm.2, lm.3, lm.4, lm.5, lm.6, lm.7, lm.8))
```

```
## [1] 52974.03 52900.42 52788.48 52803.60 52738.26 53007.42 57224.47 57698.84
```

```
# AIC
aics = AIC(lm.1, lm.2, lm.3, lm.4, lm.5, lm.6, lm.7, lm.8)[, 2]; aics
```

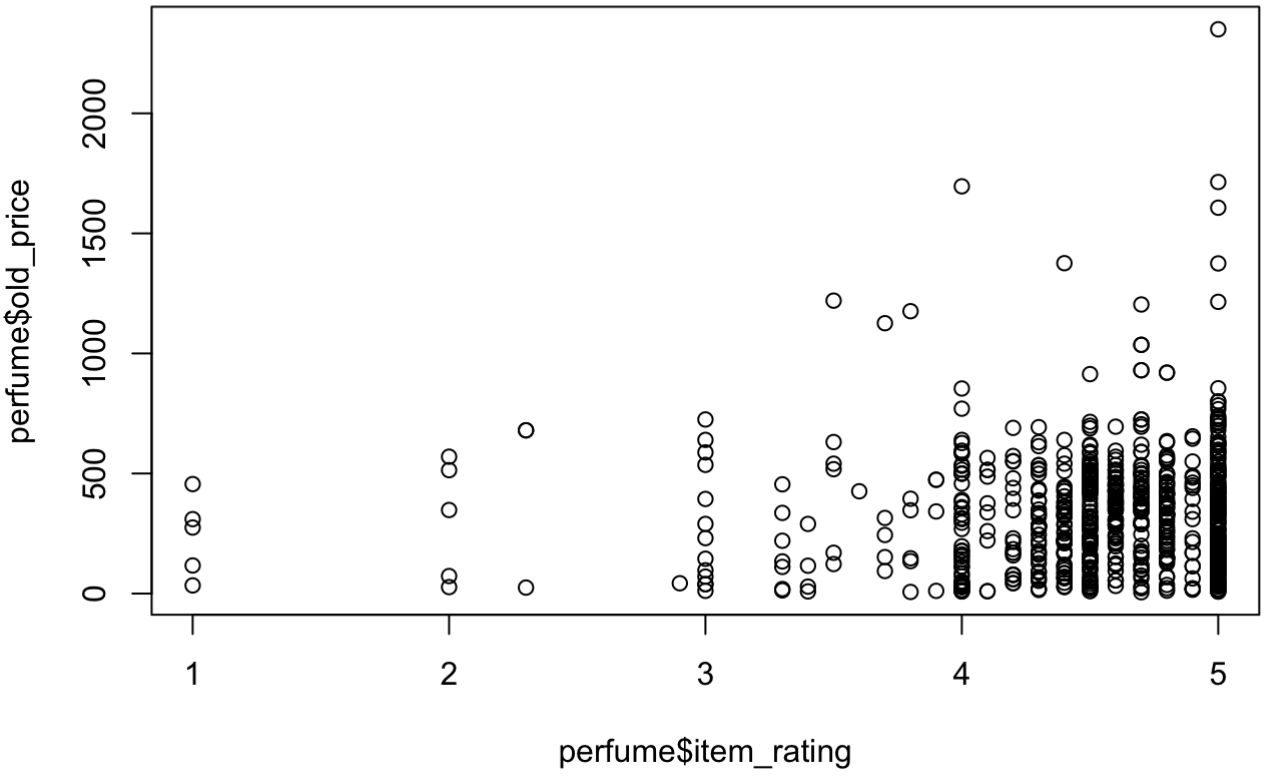
```
## [1] 10919.48 10918.37 10916.65 10916.93 10915.95 10920.21 10982.63 10988.29
```

```
# BIC
bics = BIC(lm.1, lm.2, lm.3, lm.4, lm.5, lm.6, lm.7, lm.8)[, 2]; bics
```

```
## [1] 11013.07 11007.28 11000.88 10996.48 10990.82 10981.04 11034.10 11011.68
```

```
# cannot remove scent, conc
```

```
plot(perfume$item_rating, perfume$old_price)
```




```
lm.2 = lm(item_rating ~ big_brand + is_noon + nose_rating + old_price +
            department + conc + ml_order +
            seller_rating + scent + num_sel_ratings, data = perfume)
summary(lm.2)
```

```
##
## Call:
## lm(formula = item_rating ~ big_brand + is_noon + nose_rating +
##     old_price + department + conc + ml_order + seller_rating +
##     scent + num_sel_ratings, data = perfume)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4686 -0.1553  0.1002  0.3573  0.7068
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       3.815e+00  8.819e-01   4.325 1.72e-05 ***
## big_brand         1.788e-02  4.199e-02   0.426   0.6704
## is_noon           2.207e-02  2.435e-01   0.091   0.9278
## nose_rating      -5.253e-04  7.630e-04  -0.688   0.4914
## old_price         4.640e-05  8.864e-05   0.523   0.6008
## departmentMen    -4.798e-01  5.698e-01  -0.842   0.4000
## departmentUnisex -5.048e-01  5.736e-01  -0.880   0.3791
## departmentWomen  -3.706e-01  5.668e-01  -0.654   0.5134
## concEDP           4.886e-01  5.674e-01   0.861   0.3894
## concEDT           3.883e-01  5.658e-01   0.686   0.4928
## ml_order         -3.813e-03  9.154e-03  -0.417   0.6772
## seller_rating     1.977e-01  1.024e-01   1.931   0.0539 .
## scentFloral      -7.436e-02  7.826e-02  -0.950   0.3423
## scentFresh       -1.251e-01  1.108e-01  -1.129   0.2594
## scentFruity      -2.548e-02  9.784e-02  -0.260   0.7946
## scentOriental    -5.623e-02  9.272e-02  -0.606   0.5444
## scentSpicy        5.731e-02  8.634e-02   0.664   0.5070
## scentWoody       -3.487e-02  7.916e-02  -0.440   0.6598
## num_sel_ratings  -7.176e-07  2.515e-06  -0.285   0.7755
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5621 on 777 degrees of freedom
## Multiple R-squared:  0.03631,    Adjusted R-squared:  0.01398
## F-statistic: 1.626 on 18 and 777 DF,  p-value: 0.048
```

```
lm.3 = lm(item_rating ~ old_price, data = perfume)
summary(lm.3)
```

```
##
## Call:
## lm(formula = item_rating ~ old_price, data = perfume)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5465 -0.1295  0.0735  0.4343  0.5152
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.484e+00  3.369e-02 133.071   <2e-16 ***
## old_price   1.379e-04  8.251e-05   1.671   0.0951 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5654 on 794 degrees of freedom
## Multiple R-squared:  0.003505,   Adjusted R-squared:  0.00225
## F-statistic: 2.793 on 1 and 794 DF,  p-value: 0.0951
```

```
lm.4 = lm(old_price ~ item_rating, data = perfume)
summary(lm.4)
```

```
##
## Call:
## lm(formula = old_price ~ item_rating, data = perfume)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -331.72 -184.13  -12.51  131.31 2009.78
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   213.13      69.42   3.070  0.00221 **
## item_rating    25.42      15.21   1.671  0.09510 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 242.8 on 794 degrees of freedom
## Multiple R-squared:  0.003505,   Adjusted R-squared:  0.00225
## F-statistic: 2.793 on 1 and 794 DF,  p-value: 0.0951
```