Демонстрация регрессионного анализа в R

Комков Степан

5 марта 2017 г

Данные

Для начала подключим пакет, содержащий данные, а так же все пакеты, которые понадобятся нам при работе:

```
library("ggplot2")
library("dplyr")
library("dummies")
library("caret")
library("car")
library("lmtest")
library("plm")
library("ModelMetrics")
library("erer")
library("AUC")
library("memisc")
```

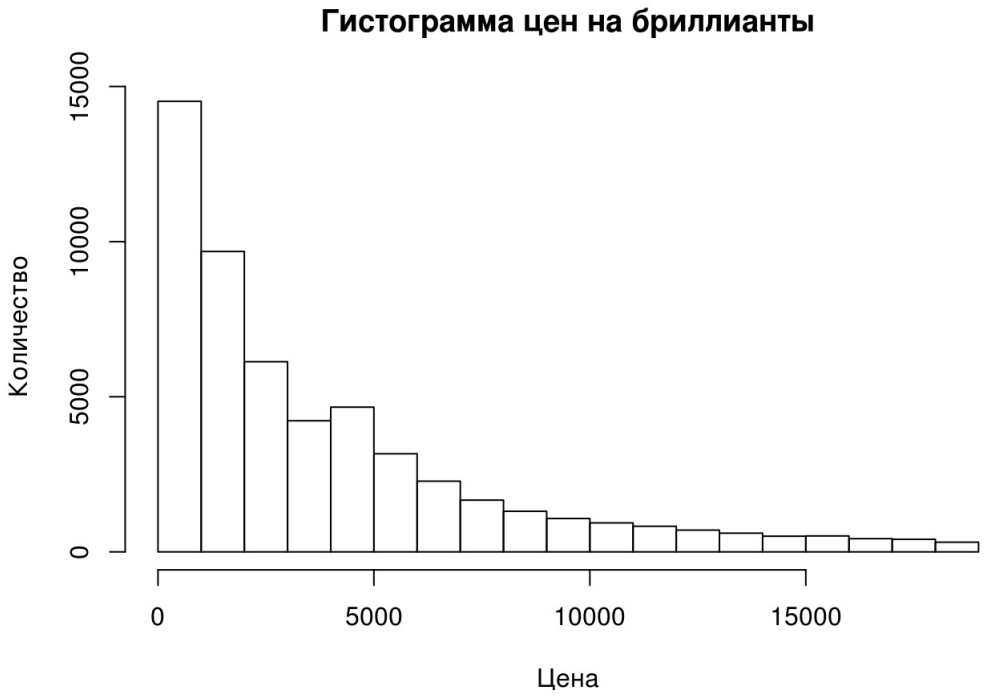
Исследовать будем базу по проданным бриллиантам из пакета ggplot2. Посмотрим на типы данных в базе:

```
str(db)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    53940 obs. of  10 variables:
##  $ carat  : num  1.14 1.2 1.51 0.95 1.24 1.01 0.39 0.41 0.34 1.02 ...
##  $ cut    : Ord.factor w/ 5 levels "Fair"<"Good"<..: 4 3 4 5 3 5 5 5 5 4 ...
##  $ color  : Ord.factor w/ 7 levels "D"<"E"<"F"<"G"<..: 1 1 4 2 5 4 4 4 2 1 ...
##  $ clarity: Ord.factor w/ 8 levels "I1"<"SI2"<"SI1"<..: 2 2 3 2 5 4 3 4 7 1 ...
##  $ depth  : num  62.6 63.2 62.7 62.2 62.6 59.6 61 61.4 62 61.4 ...
##  $ table  : num  58 60 58 56 54 59 55 54 56 60 ...
##  $ price  : int  4520 5174 10951 4068 7092 6295 675 1061 1084 3838 ...
##  $ x      : num  6.64 6.65 7.3 6.26 6.84 6.52 4.75 4.84 4.52 6.47 ...
##  $ y      : num  6.58 6.54 7.25 6.35 6.88 6.57 4.82 4.77 4.48 6.42 ...
##  $ z      : num  4.14 4.17 4.56 3.92 4.29 3.9 2.92 2.95 2.79 3.96 ...
```

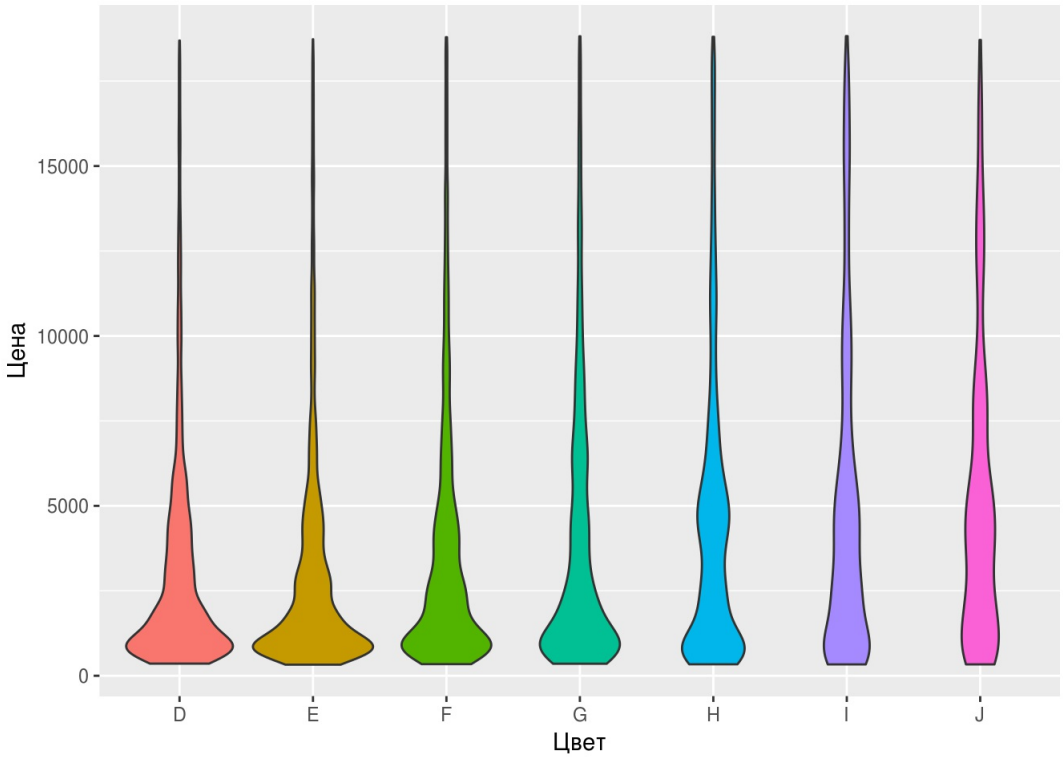
price - цена бриллианта в долларах, carat - вес, cut - качество огранки, color - цвет, clarity - прозрачность, x - длина, y - ширина, z - глубина, depth - процент глубины алмаза, table - ширина верхней поверхности относительно наибольшей ширины. Посмотрим гистограмму цен проданных бриллиантов:

```
hist(db$price,xlab="Цена",ylab="Количество",main="Гистограмма цен на бриллианты")
```



Посмотрим на плотности распределения цен в зависимости от цвета бриллианта:

```
ggplot(db,aes(color,price,fill=color))+geom_violin()+labs(x="Цвет",y="Цена")+guides(fill=FALSE)
```



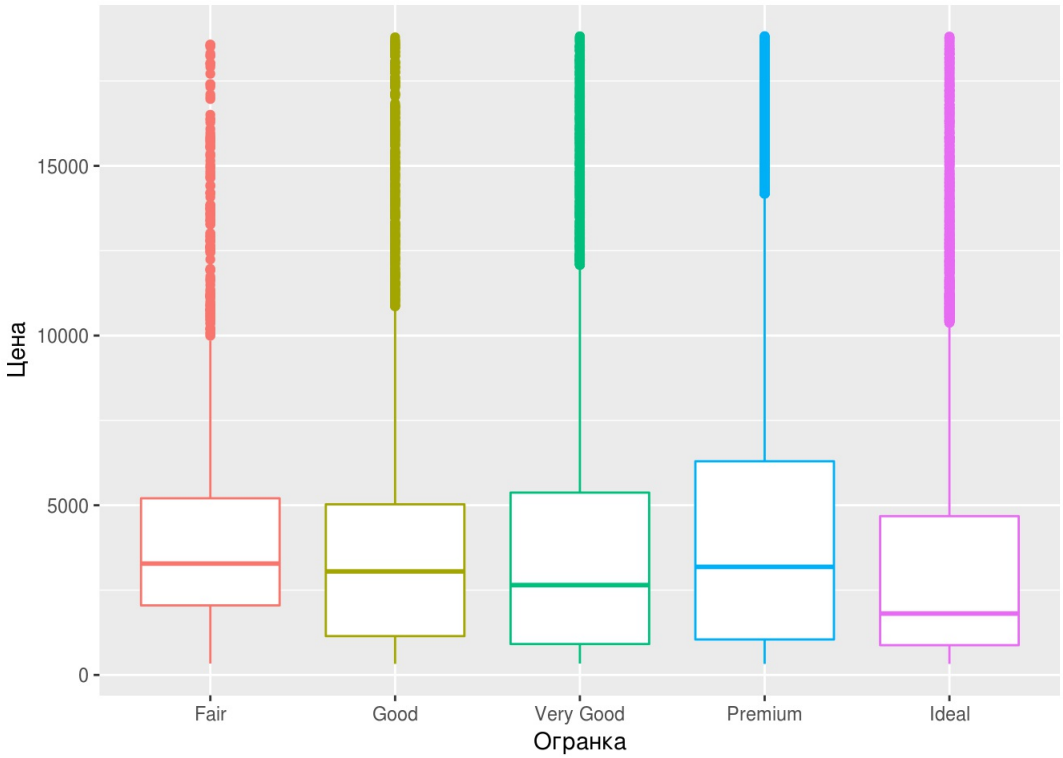
Плотности для цветов D и E визуально похожи. Давайте проверим гипотезу о равенстве средних в этих двух группах:

```
summary(aov(price~color,db1))
```

```
##                Df    Sum Sq  Mean Sq F value Pr(>F)
## color           1 3.479e+07 34791601   3.102 0.0782 .
## Residuals   16570 1.859e+11 11217453
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Итак, на уровне значимости в 5% мы не отвергаем гипотезу о том, что средние цены бриллиантов цвета D и E различаются. Посмотрим теперь на распределение цен в зависимости от огранки бриллианта:

```
ggplot(db,aes(cut,price,color=cut))+geom_boxplot()+labs(x="Огранка",y="Цена")+guides(color=F)
```



По полученным графикам видно, что разброс цен у очень хороших бриллиантов больше, чем у просто хороших. Но давайте проверим гипотезу, что на самом деле дисперсии у этих двух типов бриллиантов равны:

```
bartlett.test(data=db1,price~cut)
```

```
##
##  Bartlett test of homogeneity of variances
##
## data:  price by cut
## Bartlett's K-squared = 30.523, df = 1, p-value = 3.3e-08
```

Наша гипотеза о равенстве дисперсий отвергается.

Выбор линейной модели

Перейдем к построению линейных моделей. Для этого будем рассматривать категориальные переменные (хоть они и упорядоченные) как дамми-переменные (набор переменных-индикаторов). А так же разобьем нашу выборку на тренировочную и тестовую в отношении 3 к 1. Построим линейную модель, объясняющую зависимую переменную цены через все остальные переменные. Исследуем получившуюся модель на значимость в целом:

```
summary(model)
```

```
##
## Call:
## lm(formula = price ~ ., data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21123.0   -594.9   -184.9    376.6  10672.4
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5578.570    447.148  12.476  < 2e-16 ***
## carat       11152.446     56.116 198.739  < 2e-16 ***
## depth         -60.241      5.134 -11.735  < 2e-16 ***
## table         -26.056      3.383  -7.703 1.36e-14 ***
## x            -963.278     35.537 -27.107  < 2e-16 ***
## y              11.630     19.644   0.592 0.553831
## z             -57.884     34.472  -1.679 0.093129 .
## colorD       2391.555     30.422  78.612  < 2e-16 ***
## colorE       2164.663     29.042  74.536  < 2e-16 ***
## colorF       2113.053     28.911  73.089  < 2e-16 ***
## colorG       1893.091     28.329  66.826  < 2e-16 ***
## colorH       1392.188     29.037  47.945  < 2e-16 ***
## colorI        907.315     30.661  29.592  < 2e-16 ***
## cutFair      -805.120     38.555 -20.882  < 2e-16 ***
## cutGood      -242.350     23.586 -10.275  < 2e-16 ***
## cutVeryGood  -100.989     16.490  -6.124 9.21e-10 ***
## cutPremium    -55.703     16.921  -3.292 0.000996 ***
## clarityI1   -5396.055     59.088 -91.323  < 2e-16 ***
## claritySI2  -2647.412     35.234 -75.139  < 2e-16 ***
## claritySI1  -1687.554     33.830 -49.884  < 2e-16 ***
## clarityVS2  -1089.646     33.573 -32.456  < 2e-16 ***
## clarityVS1   -775.996     34.343 -22.596  < 2e-16 ***
## clarityVVS2  -395.619     35.943 -11.007  < 2e-16 ***
## clarityVVS1  -314.891     37.712  -8.350  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1135 on 40433 degrees of freedom
## Multiple R-squared:  0.9186, Adjusted R-squared:  0.9185
## F-statistic: 1.983e+04 on 23 and 40433 DF,  p-value: < 2.2e-16
```

Как видим, модель объясняет более 90% дисперсии в данных. Все регрессоры, за исключением двух, статистически значимы. А так же сама регрессия значима в целом. Как мы помним, средние цены в группах бриллиантов цветов D и E мы считаем равными. Проверим гипотезу о том, что переменные-индикаторы этих групп входят в модель с одинаковым весом:

```
linearHypothesis(model,"colorD-colorE=0")
```

```
## Linear hypothesis test
##
## Hypothesis:
## colorD - colorE = 0
##
## Model 1: restricted model
## Model 2: price ~ carat + depth + table + x + y + z + colorD + colorE +
##     colorF + colorG + colorH + colorI + cutFair + cutGood + cutVeryGood +
##     cutPremium + clarityI1 + claritySI2 + claritySI1 + clarityVS2 +
##     clarityVS1 + clarityVVS2 + clarityVVS1
##
##   Res.Df        RSS Df Sum of Sq      F    Pr(>F)
## 1  40434 5.2268e+10
## 2  40433 5.2113e+10  1 154558803 119.92 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Наша гипотеза отвергается на любом разумном уровне значимости, а значит рассмотренные переменные вносят разный вклад в цену. Посмотрим новую модель, добавив комбинации признаков. С помощью них мы узнаем, как влияет цвет, качество и прозрачность на стоимость каждого нового грамма бриллианта, а так же нового сантиметра в каждом из трех измерений:

```
summary(model2)
```

```
##
## Call:
## lm(formula = price ~ (. - carat - depth - table - x - y - z) *
##     (carat + x + y + z) + depth + table, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -20487.7   -285.5      8.9    256.0   8662.2
##
## Coefficients:
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       20284.816    763.937  26.553  < 2e-16 ***
## colorD            -5188.501    382.767 -13.555  < 2e-16 ***
## colorE            -1898.359    363.880  -5.217 1.83e-07 ***
## colorF            -2213.043    358.334  -6.176 6.64e-10 ***
## colorG            -4312.432    339.838 -12.690  < 2e-16 ***
## colorH            -4591.175    338.439 -13.566  < 2e-16 ***
## colorI             -637.747    366.881  -1.738 0.082166 .
## cutFair           -4802.631    395.763 -12.135  < 2e-16 ***
## cutGood           -1225.745    256.756  -4.774 1.81e-06 ***
## cutVeryGood         374.580    183.592   2.040 0.041328 *
## cutPremium         -664.146    174.992  -3.795 0.000148 ***
## clarityI1          4069.287    864.006   4.710 2.49e-06 ***
## claritySI2         3922.941    619.210   6.335 2.39e-10 ***
## claritySI1         7674.584    614.505  12.489  < 2e-16 ***
## clarityVS2         3421.290    608.513   5.622 1.90e-08 ***
## clarityVS1         3837.910    623.271   6.158 7.45e-10 ***
## clarityVVS2        4404.985    663.278   6.641 3.15e-11 ***
## clarityVVS1        1834.638    667.236   2.750 0.005969 **
## carat             17043.253    470.498  36.224  < 2e-16 ***
## x                  -310.164    658.857  -0.471 0.637814
## y                 -3181.175    635.760  -5.004 5.65e-07 ***
## z                   969.771    476.855   2.034 0.041990 *
## depth              -157.958      5.128 -30.802  < 2e-16 ***
## table               -39.132      2.459 -15.912  < 2e-16 ***
## carat:colorD       1014.290    214.591   4.727 2.29e-06 ***
## x:colorD              7.319    347.040   0.021 0.983175
## y:colorD           1050.782    335.293   3.134 0.001726 **
## z:colorD             48.996    236.414   0.207 0.835819
## carat:colorE       3044.372    198.502  15.337  < 2e-16 ***
## x:colorE            206.611    283.423   0.729 0.466016
## y:colorE            320.984    271.288   1.183 0.236743
## z:colorE           -522.461    207.570  -2.517 0.011838 *
## carat:colorF       2860.912    190.011  15.057  < 2e-16 ***
## x:colorF           -344.519    305.796  -1.127 0.259906
## y:colorF           1176.502    294.471   3.995 6.47e-05 ***
## z:colorF           -921.815    209.199  -4.406 1.05e-05 ***
## carat:colorG        700.878    172.477   4.064 4.84e-05 ***
## x:colorG            -75.436    303.436  -0.249 0.803666
## y:colorG           1248.879    291.373   4.286 1.82e-05 ***
## z:colorG           -490.712    201.453  -2.436 0.014861 *
```

```
## carat:colorH       -389.594    167.071  -2.332 0.019711 *
## x:colorH            789.542    251.293   3.142 0.001680 **
## y:colorH            554.665    235.020   2.360 0.018276 *
## z:colorH           -557.184    201.879  -2.760 0.005783 **
## carat:colorI       1273.780    179.497   7.096 1.30e-12 ***
## x:colorI           -347.309    337.821  -1.028 0.303915
## y:colorI            742.385    326.689   2.272 0.023065 *
## z:colorI           -632.075    211.740  -2.985 0.002836 **
## carat:cutFair     -3491.058    184.133 -18.959  < 2e-16 ***
## x:cutFair          2432.505    378.779   6.422 1.36e-10 ***
## y:cutFair         -1584.970    376.188  -4.213 2.52e-05 ***
## z:cutFair           575.226    162.838   3.533 0.000412 ***
## carat:cutGood     -1596.328    151.520 -10.535  < 2e-16 ***
## x:cutGood           932.422    247.811   3.763 0.000168 ***
## y:cutGood          -464.571    244.470  -1.900 0.057399 .
## z:cutGood          -145.799    133.414  -1.093 0.274475
## carat:cutVeryGood   -38.729    114.738  -0.338 0.735708
## x:cutVeryGood       113.473    243.737   0.466 0.641536
## y:cutVeryGood       -80.561    227.443  -0.354 0.723190
## z:cutVeryGood      -205.322    134.184  -1.530 0.125986
## carat:cutPremium   -943.111    105.671  -8.925  < 2e-16 ***
## x:cutPremium       1348.686    190.653   7.074 1.53e-12 ***
## y:cutPremium      -1155.169    185.627  -6.223 4.92e-10 ***
## z:cutPremium         49.221    118.462   0.415 0.677779
## carat:clarityI1   -8185.447    499.278 -16.395  < 2e-16 ***
## x:clarityI1       -4372.370    817.457  -5.349 8.90e-08 ***
## y:clarityI1        4131.381    785.750   5.258 1.46e-07 ***
## z:clarityI1          22.845    438.564   0.052 0.958458
## carat:claritySI2  -3009.199    455.496  -6.606 3.99e-11 ***
## x:claritySI2      -4427.319    617.241  -7.173 7.48e-13 ***
## y:claritySI2       3773.133    598.326   6.306 2.89e-10 ***
## z:claritySI2       -287.761    429.155  -0.671 0.502525
## carat:claritySI1    282.194    457.319   0.617 0.537198
## x:claritySI1      -3868.535    621.455  -6.225 4.86e-10 ***
## y:claritySI1       2608.040    602.855   4.326 1.52e-05 ***
## z:claritySI1       -912.876    428.843  -2.129 0.033286 *
## carat:clarityVS2  -1185.080    456.077  -2.598 0.009369 **
## x:clarityVS2      -2402.635    624.064  -3.850 0.000118 ***
## y:clarityVS2       1895.311    606.026   3.127 0.001765 **
## z:clarityVS2       -446.712    429.277  -1.041 0.298060
## carat:clarityVS1   -260.118    465.311  -0.559 0.576151
## x:clarityVS1      -3623.625    618.376  -5.860 4.67e-09 ***
## y:clarityVS1       2854.749    598.275   4.772 1.83e-06 ***
## z:clarityVS1       -258.995    434.164  -0.597 0.550821
## carat:clarityVVS2  1514.627    500.202   3.028 0.002463 **
## x:clarityVVS2     -2362.274    667.283  -3.540 0.000400 ***
## y:clarityVVS2      1957.780    650.016   3.012 0.002598 **
## z:clarityVVS2     -1207.772    467.984  -2.581 0.009861 **
## carat:clarityVVS1   307.066    509.054   0.603 0.546373
## x:clarityVVS1     -1190.263    718.231  -1.657 0.097484 .
## y:clarityVVS1      1015.524    701.041   1.449 0.147459
## z:clarityVVS1      -476.348    480.159  -0.992 0.321173
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 811.2 on 40365 degrees of freedom
## Multiple R-squared:  0.9585, Adjusted R-squared:  0.9584
## F-statistic: 1.024e+04 on 91 and 40365 DF,  p-value: < 2.2e-16
```

Как мы видим, модель стала объяснять большую часть дисперсии, но теперь много предикторов статистически незначимы. Давайте сраним две имеющиеся модели:

```
mtable(model, model2)$summaries
```

```
##                model           model2
## R-squared      "0.9"           "1.0"
## adj. R-squared "0.9"           "1.0"
## sigma          "1135.3"        "811.2"
## F              "19829.6"       "10244.5"
## p              "0.0"           "0.0"
## Log-likelihood "-341994.6"     "-328359.4"
## Deviance       "52113405976.7" "26558945328.2"
## AIC            "684039.1"      "656904.8"
## BIC            "684254.3"      "657705.4"
## N              "40457"         "40457"
```

Хоть во второй модели значительно больше регрессоров, штрафные значения Акаике и Шварца у нее меньше. Давайте проверим гипотезу о том, что все добавленные регрессоры на самом деле лишние с помощью теста Вальда:

```
waldtest(model,model2)
```

```
## Wald test
##
## Model 1: price ~ carat + depth + table + x + y + z + colorD + colorE +
##     colorF + colorG + colorH + colorI + cutFair + cutGood + cutVeryGood +
##     cutPremium + clarityI1 + claritySI2 + claritySI1 + clarityVS2 +
##     clarityVS1 + clarityVVS2 + clarityVVS1
## Model 2: price ~ ((carat + depth + table + x + y + z + colorD + colorE +
##     colorF + colorG + colorH + colorI + cutFair + cutGood + cutVeryGood +
##     cutPremium + clarityI1 + claritySI2 + claritySI1 + clarityVS2 +
##     clarityVS1 + clarityVVS2 + clarityVVS1) - carat - depth -
##     table - x - y - z) * (carat + x + y + z) + depth + table
##   Res.Df Df      F    Pr(>F)
## 1  40433
## 2  40365 68 571.15 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Как показывает пи-значение, гипотеза отвергается на любом разумном уровне значимости. Чтобы хоть как-то уменьшить количество переменных во второй модели, построим новую модель, пошагово убирая предикторы, руководствуясь значением Акаике. Посмотрим на оставшиеся регрессоры:

```
step_model <- step(model2,direction="backward",trace=0)
formula(step_model)
```

```
## price ~ colorD + colorE + colorF + colorG + colorH + colorI +
##     cutFair + cutGood + cutVeryGood + cutPremium + clarityI1 +
##     claritySI2 + claritySI1 + clarityVS2 + clarityVS1 + clarityVVS2 +
##     clarityVVS1 + carat + x + y + z + depth + table + colorD:carat +
##     colorD:y + colorE:carat + colorE:x + colorE:z + colorF:carat +
##     colorF:y + colorF:z + colorG:carat + colorG:y + colorG:z +
##     colorH:carat + colorH:x + colorH:y + colorH:z + colorI:carat +
##     colorI:y + colorI:z + cutFair:carat + cutFair:x + cutFair:y +
##     cutFair:z + cutGood:carat + cutGood:x + cutGood:y + cutGood:z +
##     cutVeryGood:z + cutPremium:carat + cutPremium:x + cutPremium:y +
##     clarityI1:carat + clarityI1:x + clarityI1:y + claritySI2:carat +
##     claritySI2:x + claritySI2:y + claritySI2:z + claritySI1:carat +
##     claritySI1:x + claritySI1:y + claritySI1:z + clarityVS2:carat +
##     clarityVS2:x + clarityVS2:y + clarityVS2:z + clarityVS1:x +
##     clarityVS1:y + clarityVS1:z + clarityVVS2:carat + clarityVVS2:x +
##     clarityVVS2:y + clarityVVS2:z + clarityVVS1:carat + clarityVVS1:x +
##     clarityVVS1:y + clarityVVS1:z
```

Исследование модели

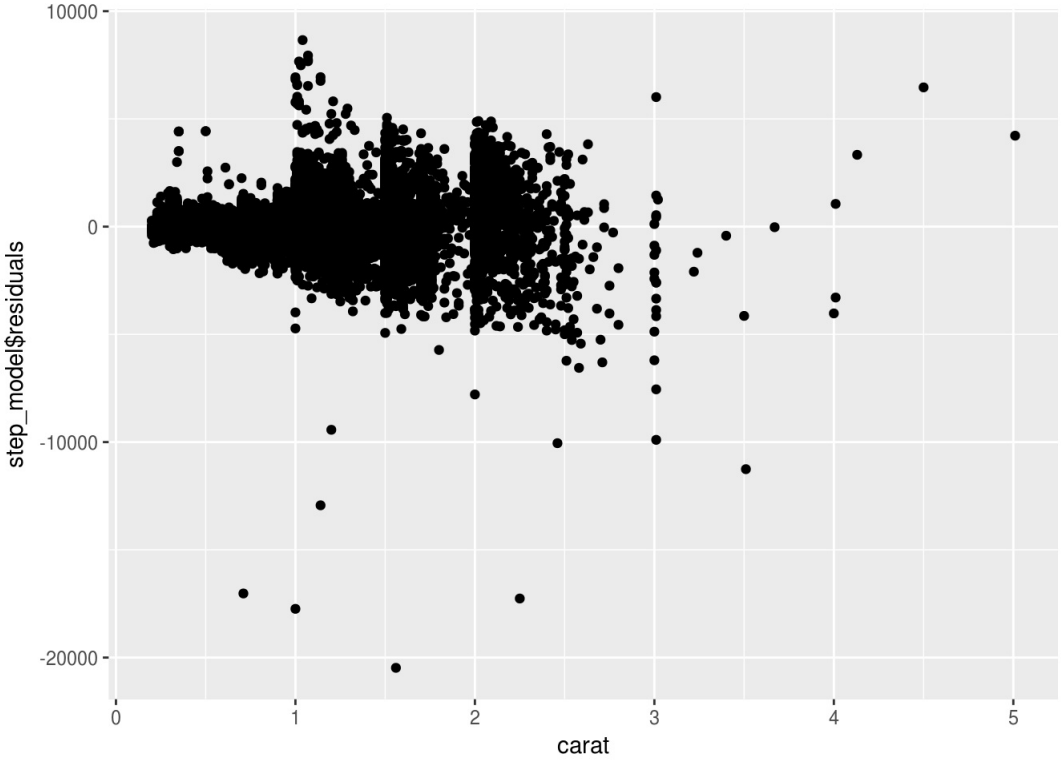
При построении моделей оценки одних и тех же коэффициентов сильно менялись. Скорее всего среди наших признаков имеется сильная мультиколлениарность. Посмотрим на коэффициенты вздутия дисперсии:

```
vif(step_model)
```

```
##            colorD            colorE            colorF            colorG
##      8.007319e+02      1.062677e+03      1.014207e+03      1.018522e+03
##            colorH            colorI           cutFair          cutGood
##      7.978981e+02      6.655063e+02      2.705553e+02      3.023494e+02
##       cutVeryGood       cutPremium        clarityI1        claritySI2
##      3.473069e+01      2.854386e+02      3.497662e+02      6.447704e+02
##        claritySI1        clarityVS2        clarityVS1       clarityVVS2
##      7.748854e+02      6.803103e+02      1.653210e+02      7.030950e+02
##       clarityVVS1             carat                 x                 y
##      5.265777e+02      4.526806e+02      2.589054e+04      2.724536e+04
##                 z             depth             table      colorD:carat
##      8.010128e+02      3.131552e+00      1.842384e+00      1.473665e+02
##          colorD:y      colorE:carat          colorE:x          colorE:z
##      1.467115e+03      1.952479e+02      2.702304e+03      2.094405e+03
##      colorF:carat          colorF:y          colorF:z      colorG:carat
##      2.110585e+02      2.912308e+03      2.357722e+03      2.227877e+02
##          colorG:y          colorG:z      colorH:carat          colorH:x
##      3.005187e+03      2.418439e+03      2.242067e+02      7.254537e+03
##          colorH:y          colorH:z      colorI:carat          colorI:y
##      6.965695e+03      2.138713e+03      2.275189e+02      2.432981e+03
##          colorI:z     cutFair:carat         cutFair:x         cutFair:y
##      1.940251e+03      7.703609e+01      9.572324e+03      9.289318e+03
##         cutFair:z     cutGood:carat         cutGood:x         cutGood:y
##      5.224691e+02      9.749283e+01      9.858481e+03      9.823384e+03
##         cutGood:z     cutVeryGood:z  cutPremium:carat      cutPremium:x
##      6.624271e+02      3.636323e+01      1.130452e+02      1.169184e+04
##      cutPremium:y   clarityI1:carat       clarityI1:x       clarityI1:y
##      1.156882e+04      1.035168e+02      2.477015e+04      2.330629e+04
##  claritySI2:carat      claritySI2:x      claritySI2:y      claritySI2:z
##      2.856922e+02      1.274067e+05      1.272269e+05      2.052494e+03
##  claritySI1:carat      claritySI1:x      claritySI1:y      claritySI1:z
##      2.643285e+02      1.438508e+05      1.424252e+05      2.533103e+03
##  clarityVS2:carat      clarityVS2:x      clarityVS2:y      clarityVS2:z
##      2.126218e+02      1.294175e+05      1.285330e+05      2.370580e+03
##      clarityVS1:x      clarityVS1:y      clarityVS1:z clarityVVS2:carat
##      8.697728e+04      8.698182e+04      1.874641e+03      1.736790e+02
##     clarityVVS2:x     clarityVVS2:y     clarityVVS2:z clarityVVS1:carat
##      6.242796e+04      6.213404e+04      3.284689e+03      1.005763e+02
##     clarityVVS1:x     clarityVVS1:y     clarityVVS1:z
##      4.781539e+04      4.757874e+04      2.504094e+03
```

Как мы видим, у множества признаков рассматриваемые коэффициенты значительно превосходят 1000, что говорит о сильной мультиколлинеарности наших предикторов. Однако отсутствие мультиколлинеарности не является для нас критичным свойством. Для эффективности оценок модели нужно, чтобы выполнялись следующие свойства для ошибок: условная гомоскедостичность, условная некоррелируемость и строгая экзогенность. Давайте посмотрим, соответствует ли модель этим предпосылкам. Для наблюдения условной гетероскедастичности, если она имеется, построим график остатков модели в зависимости от веса бриллианта:

```
ggplot(aes(x=carat,y=step_model$residuals),data=train)+geom_point()
```



Как видно из графика в данных имеется явная условная гетероскедастичность. Чем больше бриллиант - тем больше варьируется его цена. Убедимся в этом, проведя тест Голдфелда—Куандта, сортируя наблюдения по весу, убирая средние 20% наблюдений:

```
gqtest(step_model,order.by = ~carat, data=train, fraction = 0.2)
```

```
##
##  Goldfeld-Quandt test
##
## data:  step_model
## GQ = 69.293, df1 = 16103, df2 = 16102, p-value < 2.2e-16
```

Как мы убедились, гипотеза о гомоскедастичности отвергается на любом разумном уровне значимости. К счастью, наши данные не имеют никакой временной или географической структуры, так что мы можем считать, что в данных нет автокорреляции. Чтобы подтвердить это, проведем тесты Дарбина-Уотсона и Бройша—Годфри второго порядка:

```
dwt(step_model)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1     0.001978006       1.99595   0.656
##  Alternative hypothesis: rho != 0
```

```
bgtest(step_model,order=2)
```

```
##
##  Breusch-Godfrey test for serial correlation of order up to 2
##
## data:  step_model
## LM test = 1.1083, df = 2, p-value = 0.5746
```

Как видим, оба теста не отвергают нашу гипотезу даже на высоких уровнях значимости. Итак, в наших данный присутствует гетероскедастичноть, но отсутствует автокорреляция. Значит для оценки ковариационной матрицы регрессоров достаточно использовать робастную оценку устойчивую к гетероскедастичности. Посмотрим, какие регрессоры значимы при таком оценивании:

```
coeftest(step_model, vcov.=vcovHC(step_model))
```

```
##
## t test of coefficients:
##
##                    Estimate Std. Error  t value  Pr(>|t|)
## (Intercept)       19841.7450  1704.2425  11.6426 < 2.2e-16 ***
## colorD            -5069.7286  1542.1400  -3.2875 0.0010118 **
## colorE            -1815.6008  1018.8600  -1.7820 0.0747580 .
## colorF            -2152.9126  1648.2662  -1.3062 0.1915029
```

```
## colorG            -4216.3222  2074.2926  -2.0327 0.0420939 *
## colorH            -4529.4423  2109.8499  -2.1468 0.0318145 *
## colorI             -581.6765  1007.9870  -0.5771 0.5638971
## cutFair           -4800.1656  6936.1222  -0.6921 0.4889079
## cutGood           -1182.6684  2069.1024  -0.5716 0.5676062
## cutVeryGood         435.4484   120.7693   3.6056 0.0003118 ***
## cutPremium         -626.5229  1234.3693  -0.5076 0.6117611
## clarityI1          4380.1063  3594.6602   1.2185 0.2230398
## claritySI2         4222.7053  2497.2139   1.6910 0.0908509 .
## claritySI1         7970.5495  1284.1764   6.2067 5.462e-10 ***
## clarityVS2         3721.3146  2338.2544   1.5915 0.1115066
## clarityVS1         4170.6028   407.0224  10.2466 < 2.2e-16 ***
## clarityVVS2        4697.6198  1258.8330   3.7317 0.0001904 ***
## clarityVVS1        2114.7110  4426.3281   0.4778 0.6328255
## carat             16757.4306   748.5905  22.3853 < 2.2e-16 ***
## x                  -428.0862  1539.2472  -0.2781 0.7809263
## y                 -2990.7796  1503.5065  -1.9892 0.0466855 *
## z                  1029.6149   543.3162   1.8951 0.0580920 .
## depth              -157.5088    22.8285  -6.8997 5.289e-12 ***
## table               -38.9426     3.5063 -11.1065 < 2.2e-16 ***
## colorD:carat       1078.9330  1059.7979   1.0181 0.3086577
## colorD:y           1058.9490   412.2096   2.5690 0.0102041 *
## colorE:carat       3077.6525   628.6934   4.8953 9.853e-07 ***
## colorE:x            520.0232   318.8641   1.6309 0.1029274
## colorE:z           -540.4392   500.1326  -1.0806 0.2798852
## colorF:carat       2877.2699  1046.6660   2.7490 0.0059807 **
## colorF:y            834.9116   338.2642   2.4682 0.0135827 *
## colorF:z           -946.6115   398.7192  -2.3741 0.0175950 *
## colorG:carat        748.1117  1302.7712   0.5742 0.5658043
## colorG:y           1153.0525   394.5520   2.9224 0.0034750 **
## colorG:z           -494.9935   454.7611  -1.0885 0.2763945
## colorH:carat       -365.9920  1222.6329  -0.2993 0.7646765
## colorH:x            960.8758   850.9423   1.1292 0.2588243
## colorH:y            369.6033   751.9183   0.4915 0.6230422
## colorH:z           -557.1577   428.4630  -1.3004 0.1934838
## colorI:carat       1289.7389   582.6704   2.2135 0.0268690 *
## colorI:y            393.2816   317.9700   1.2369 0.2161495
## colorI:z           -647.8998   518.4480  -1.2497 0.2114197
## cutFair:carat     -3484.5979  3031.0641  -1.1496 0.2503037
## cutFair:x          2453.1462  1016.5967   2.4131 0.0158220 *
## cutFair:y         -1598.4537   732.7245  -2.1815 0.0291507 *
## cutFair:z           560.9472   923.7974   0.6072 0.5437091
## cutGood:carat     -1568.9097  1200.4800  -1.3069 0.1912534
## cutGood:x           919.2015   692.9493   1.3265 0.1846796
## cutGood:y          -448.7822   654.1988  -0.6860 0.4927153
## cutGood:z          -168.5137   354.0148  -0.4760 0.6340717
## cutVeryGood:z      -179.0163    35.0368  -5.1094 3.247e-07 ***
## cutPremium:carat   -919.6290   750.0505  -1.2261 0.2201722
## cutPremium:x       1394.7700   516.9871   2.6979 0.0069811 **
## cutPremium:y      -1180.9337   483.1189  -2.4444 0.0145138 *
## clarityI1:carat   -7938.5768  2012.5105  -3.9446 8.006e-05 ***
## clarityI1:x       -4484.6303  1945.8571  -2.3047 0.0211880 *
## clarityI1:y        4169.6238  1667.5696   2.5004 0.0124086 *
## claritySI2:carat  -2772.3826  1373.3634  -2.0187 0.0435270 *
## claritySI2:x      -4518.3163  1716.3584  -2.6325 0.0084791 **
## claritySI2:y       3793.8188  1643.8588   2.3079 0.0210112 *
## claritySI2:z       -311.0502   427.1881  -0.7281 0.4665358
## claritySI1:carat    516.1104   945.0396   0.5461 0.5849825
## claritySI1:x      -3952.5405  1547.6383  -2.5539 0.0106555 *
## claritySI1:y       2621.1542  1499.5333   1.7480 0.0804751 .
## claritySI1:z       -933.4891   435.7386  -2.1423 0.0321741 *
## clarityVS2:carat   -948.5578  1528.7746  -0.6205 0.5349523
## clarityVS2:x      -2496.1231  1584.6707  -1.5752 0.1152254
## clarityVS2:y       1923.6212  1528.0343   1.2589 0.2080787
## clarityVS2:z       -478.2976   440.9737  -1.0846 0.2780877
## clarityVS1:x      -3804.9994  1547.4457  -2.4589 0.0139409 *
## clarityVS1:y       2983.9182  1513.9964   1.9709 0.0487434 *
## clarityVS1:z       -326.2932   642.7870  -0.5076 0.6117208
## clarityVVS2:carat  1744.0237   963.7436   1.8096 0.0703599 .
## clarityVVS2:x     -2444.2415  1543.7544  -1.5833 0.1133587
## clarityVVS2:y      1983.0665  1512.6107   1.3110 0.1898576
## clarityVVS2:z     -1249.5191   450.6833  -2.7725 0.0055653 **
## clarityVVS1:carat   527.1408  3481.7939   0.1514 0.8796616
## clarityVVS1:x     -1257.0538  1704.6654  -0.7374 0.4608715
## clarityVVS1:y      1035.8218  1609.2984   0.6436 0.5198073
## clarityVVS1:z      -528.9659   705.4657  -0.7498 0.4533730
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

По крайней мере вес бриллианта остается статистически значимым, что соответствует простой логике. Одной из предпосылок эндогенности является пропуск регрессора. Наличие пропущенных регрессоров мы можем выявить с помощью теста Рамсея:

```
resettest(step_model)
```

```
##
##  RESET test
##
## data:  step_model
## RESET = 4775, df1 = 2, df2 = 40375, p-value < 2.2e-16
```

Гипотеза о наличии всех необходимых регрессоров отвергается на любом разумном уровне значимости. А значит, скорее всего, в наших данных присутствует эндогенность. Таким образом, наши оценки неэффективны, однако это не является помехой для использования модели в качестве предсказывающего алгоритма.

Предсказания

Предскажем стоимость каждого бриллианта из тестовой выборки с помощью всех трех построенных моделей. Посмотрим на корень из среднеквадратичной ошибки для каждой модели:

```
mse(test$price,original_rez)**0.5
```

```
## [1] 1114.921
```

```
mse(test$price,poly_rez)**0.5
```

```
## [1] 854.6674
```

```
mse(test$price,step_rez)**0.5
```

```
## [1] 852.0077
```

Наша финальная модель показывает результат схожий со второй моделью, но явно лучше чем у оригинальной модели.

Логистическая регрессия

Воспользуемся данными по выборам президента США 96-го года.

```
str(db)
```

```
## 'data.frame':    708 obs. of  8 variables:
##  $ popul : int  31 22 87 50 9 75 0 170 42 15 ...
##  $ TVnews: int  2 7 4 4 7 4 1 2 0 7 ...
##  $ ClinLR: int  -2 -1 0 -2 -3 -2 -2 0 2 0 ...
##  $ DoleLR: int  2 0 2 2 0 3 1 0 1 3 ...
##  $ age   : int  36 47 41 44 79 62 35 53 40 51 ...
##  $ educ  : int  2 -1 0 -1 3 2 1 -1 -1 0 ...
##  $ income: num  82.5 27.5 32.5 55 32.5 ...
##  $ vote  : Factor w/ 2 levels "0","1": 2 1 1 1 2 2 2 2 1 1 ...
```

popul - популяция в городе респондента, TVnews - количество дней недели, в которые респондент смотрит новости, ClinLR и DoleLR - насколько либерально или консервативно оценивает респондент программу соответствующего кандидата в президенты, age - возраст, educ - уровень образования (от неоконченного среднего до высшего специального), income - заработок, vote - голос. Разделим снова все наблюдения на тренировочную и тестовую выборки в отношении 3 к 1 и построим логистическую регрессию для предсказания голоса респондента на выборах:

```
summary(logit_model)
```

```
##
## Call:
## glm(formula = vote ~ ., family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2861  -0.8598  -0.3742   0.8922   3.0507
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5737842  0.3930180  -4.004 6.22e-05 ***
## popul       -0.0001937  0.0001237  -1.565 0.117547
## TVnews      -0.0542059  0.0424877  -1.276 0.202026
## ClinLR      -1.1160697  0.1147820  -9.723  < 2e-16 ***
## DoleLR      -0.3638688  0.0961051  -3.786 0.000153 ***
## age          0.0047912  0.0070312   0.681 0.495609
## educ         0.0029622  0.0741296   0.040 0.968126
## income       0.0077231  0.0037112   2.081 0.037430 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 721.52  on 531  degrees of freedom
## Residual deviance: 549.05  on 524  degrees of freedom
## AIC: 565.05
##
## Number of Fisher Scoring iterations: 5
```

Итак, уровень образования и просмотр новостей не являются статистически значимыми. Построим логистическую регрессию без этих предикторов:

```
summary(logit_model2)
```

```
##
## Call:
## glm(formula = vote ~ . - educ - TVnews, family = binomial(link = "logit"),
##     data = train, x = T)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3319  -0.8689  -0.3665   0.8887   3.0026
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.6229210  0.3920063  -4.140 3.47e-05 ***
## popul       -0.0001947  0.0001243  -1.567 0.117221
## ClinLR      -1.1120522  0.1144954  -9.713  < 2e-16 ***
## DoleLR      -0.3610502  0.0949262  -3.803 0.000143 ***
## age          0.0014248  0.0064341   0.221 0.824743
## income       0.0079535  0.0034764   2.288 0.022146 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 721.52  on 531  degrees of freedom
## Residual deviance: 550.69  on 526  degrees of freedom
## AIC: 562.69
##
## Number of Fisher Scoring iterations: 5
```

Проверим гипотезу о том, что наши ограничения верны:

```
lrtest(logit_model,logit_model2)
```

```
## Likelihood ratio test
##
## Model 1: vote ~ popul + TVnews + ClinLR + DoleLR + age + educ + income
## Model 2: vote ~ (popul + TVnews + ClinLR + DoleLR + age + educ + income) -
##     educ - TVnews
##   #Df  LogLik Df  Chisq Pr(>Chisq)
## 1   8 -274.53
## 2   6 -275.35 -2 1.6414     0.4401
```

Как видно, гипотеза не отвергается даже на высоких уровнях значимости. Давайте посмотрим, как выглядит средний респондент:

```
summary(train)[4,]
```

```
##               popul              TVnews              ClinLR
##  "Mean   : 283.8  "    "Mean   :3.66  "  "Mean   :-1.041  "
##              DoleLR                 age                educ
##  "Mean   : 1.361  "   "Mean   :46.46  " "Mean   : 0.4981  "
##              income                vote
##  "Mean   : 44.94  "                  NA
```

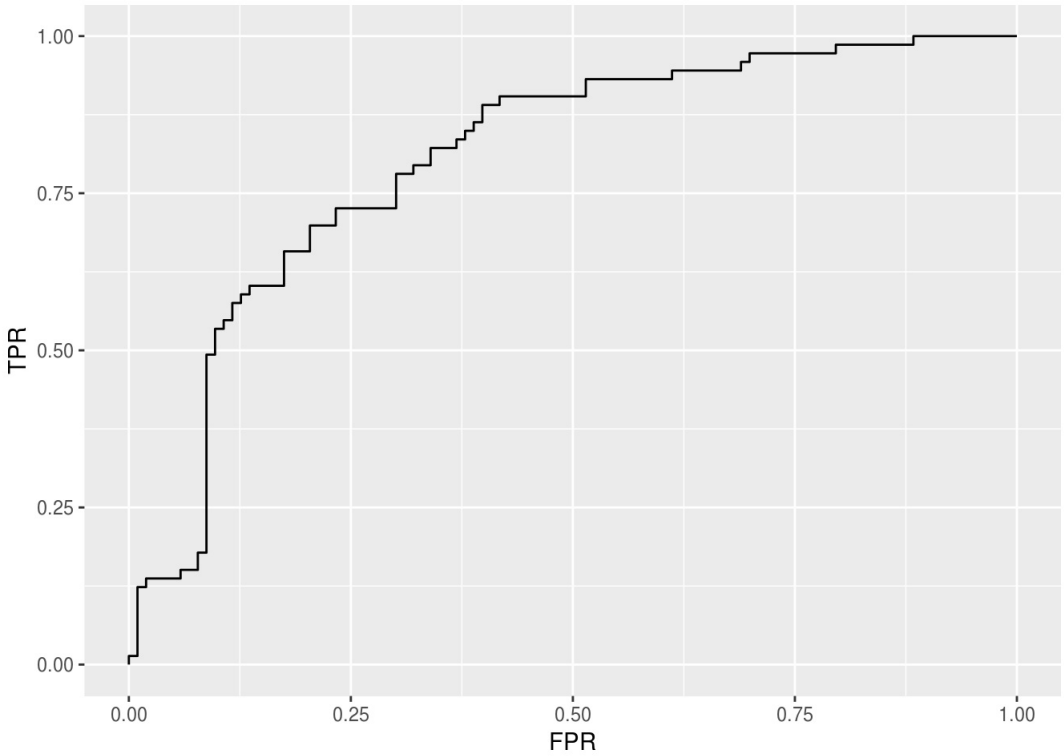
И посмотрим на предельные эффекты признаков для этого респондента:

```
maBina(logit_model2,x.mean=T)
```

```
##             effect error t.value p.value
## (Intercept) -0.373 0.086  -4.316   0.000
## popul        0.000 0.000  -1.569   0.117
## ClinLR      -0.255 0.025 -10.376   0.000
## DoleLR      -0.083 0.022  -3.809   0.000
## age          0.000 0.001   0.221   0.825
## income       0.002 0.001   2.292   0.022
```

Наконец, предскажем голоса избирателей из тестовой выборки. Построим для полученных значений ROC кривую:

```
qplot(x = roc.data$fpr, y = roc.data$tpr, geom = "line")+xlab('FPR')+ylab('TPR')
```



Площадь под ROC кривой:

```
auc(roc.data)
```

```
## [1] 0.8047613
```

А так же для различных порогов выведем процент совпадения предсказанных голосов с настоящими голосами:

```r
for(i in seq(0.1,0.9,0.05)){
  pred_vote <- as.numeric(pred>i)
  cat(i,sum(pred_vote==test$vote)/nrow(test),'\n')
}
```

```
## 0.1 0.5454545
## 0.15 0.5965909
## 0.2 0.6363636
## 0.25 0.6704545
## 0.3 0.7102273
## 0.35 0.7159091
## 0.4 0.7272727
## 0.45 0.7272727
## 0.5 0.7443182
## 0.55 0.75
## 0.6 0.7443182
## 0.65 0.6818182
## 0.7 0.6079545
## 0.75 0.6079545
## 0.8 0.625
## 0.85 0.5965909
## 0.9 0.5909091
```