Content

Final Project Report for Regression Analysis

Team member: Xiaotian Ding, Ray Liu, Jie Gu, De Lu

I. Introduction

Real estate is an industry full of opportunities and risks, while residential is the most concerned category of real estate. Therefore, it's practical to conduct research on residential building dataset.

The residential building dataset is comprised of 8 project physical and financial variables (V1~V8), 19 economic variables and indices in 5 time-lag numbers (V11~V29). The two output variables are construction costs (V9) and sale price (V10). There are 372 observations, which should be a small-sized dataset.

II. Model Building

i. Data Preprocessing

One of the Project Physical and Financial Variables is pretty special - V1 zip code, ranging from 1 to 20. Since situations may differ between locations, we defined V1 as a categorical variable. As we can see from the summary of the full model, the zip codes' P-values are large, so in the following steps, we did not use this variable into regression.

In order to apply k-fold cross-validation to check the model and its predictive ability in the final step, we split the dataset into two parts. By selecting 38 observations out randomly, the original database was split into training dataset (with 334 observations) and testing dataset (with 38 observations). In fact, we repeated the random selection performance for several times and found that the final models are almost indistinguishable.

Additionally, correlation transformation was applied to the remaining 26 predictors to help with controlling round off errors. Expressing the regression coefficients in the same units may be of help when these coefficients are compared. The standardization involves centering and scaling is as follows:

$$\frac{X_{ik} - \bar{X}_k}{sd_k} \quad (k = 1, \ldots, 26)$$

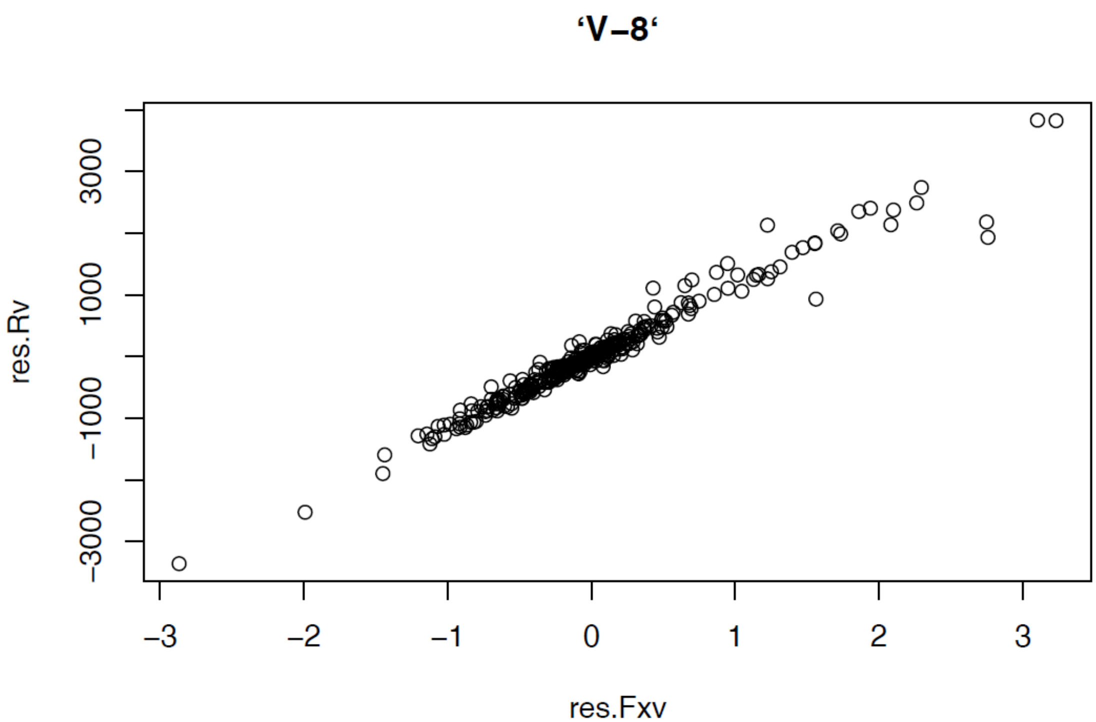
ii. Model Selection

Since there are so many as 27 variables in the original dataset, if we just build a simple linear regression model with all the variables, there must be some useless variables and multicollinearity. In order to improve efficiency and reduce noise, Lasso is the first choice to preliminarily screen the necessary predictors. We first study the regression formation of output V9.

Use the most important variable selected by Lasso as the variable of Model1， then here comes the basic model of V9 - with the most important variables in the first-order form.

$$E\{Y_i\} = \beta_0 + \beta_1 X_{i4} + \beta_2 X_{i7} + \beta_3 X_{i8} + \beta_4 X_{i16} + \beta_5 X_{i17} + \beta_6 X_{i18} + \beta_7 X_{i20} + \beta_8 X_{i21} + \beta_9 X_{i23} + \beta_{10} X_{i26} + \beta_{11} X_{i28} + \beta_{12} X_{i29}$$

Although we have sorted out the 12 most important variables, they are not necessarily linear. Drawing added-variable plots is a good method to determine the exact functional form of an independent variable in a multiple regression model. So the next step is to draw the added-variable plot of each variable in Model 1 against the other variables. Then we got 12 added-variable plots listed below.



As can be seen from the first plot, the added-variable plot of V8 is exactly linear, however, the shape in the other plots seem curved. Then we decided to add the quadratic variables into model 1 then use Lasso once again to sort out the Model 2.

Then here comes our Model 2 of V9 - with quadratic variables.

$$E\{Y_i\} = \beta_0 + \beta_1 X_{i4} + \beta_2 X_{i7} + \beta_3 X_{i7}^2 + \beta_4 X_{i8} + \beta_5 X_{i17}^2 + \beta_6 X_{i18}^2 + \beta_7 X_{i20}^2 + \beta_8 X_{i23} + \beta_9 X_{i23}^2$$

iii. Diagnostics

We use the Brown-Forsythe Test to determine whether the errors have constant variance. Since there are a series of predictors in the dataset, it's impractical to use the median of X values to divide the observations into 2 groups. Hence, we did an innovation to the original method - use the median of Y values to divide the observations. The test outcome showed that the error variance of Model 1 is not constant.

$$|t_{BF}^*| = 5.81 > t\left(1 - \frac{.05}{2}; n - p - 1\right) = 1.97$$

Therefore, the Weighted Least Squares Estimation is necessary to remedy the unequal error variances. By calculating the weighted least square diagonal matrix $W_{n\times n}^{1/2}$, we got the weighted regression Model 1, which is stated below.

$$\mathrm{Y}_w = \mathrm{X}_w \beta + \varepsilon_w$$

After the weighted transformation, we use the Brown-Forsythe Test again to test the effect of Weighted Least Squares Estimation. The new P-value is 0.15, and now, the errors' variances of Model 1 are constant. Similarly, the errors' variances of Model 2 are not constant, either. We did the same operation on Model 2, making Model 2 also satisfactory.

Outliers will affect the accuracy of our models. Bonferroni critical value of t[1-.05/(2n); n-p-1] is used to determine Y outliers. Cook's distant with percentile value larger than 20% and DFFITS with the absolute value larger than $2 \times \sqrt{\frac{p}{n}}$ are used to determine influential X outliers.

The Variance Inflation Factor is a formal measure of multicollinearity. A large VIF indicates the existence of multicollinearity. We calculated every variable in the models. The results show that our models do have slight multicollinearity, but the accuracy of the prediction will not be affected.

iv. Final Model

After removing the Y and influential X outliers, we refitted the coefficient value with the same terms in the two models. We used the Coefficient of Determination (R^2) to determine whether our models did a good job. To verify that our models don't have an overfitting problem, we recalculated the coefficients on the testing dataset with the exact same terms. The results show

that our model has a high R^2 in both the training dataset and the testing dataset, and our models all performed well.
We performed the same regression process for output V10 as V9. The adjusted R-squared values of models are as below.

	Output V9				**Output V10**			
	Model 1		**Model 2**		**Model 3**		**Model 4**	
	training	**testing**	**training**	**testing**	**training**	**testing**	**training**	**testing**
R^2	99.74%	99.87%	99.55%	99.62%	97.68%	97.98%	88.05%	95.17%
R_a^2	99.73%	99.79%	99.54%	99.45%	97.65%	97.71%	88.01%	95.03%

By comparison, we decided to choose Model 1 and Model 3 as our final model. Because it has a higher R^2 value, the form is also more simple, reducing unnecessary noise.

The final model for output V9 is:

$$E\{Y_i\} = 183.507 + 12.823X_{i4} + 82.864X_{i7} + 1183.677X_{i8} - 12.875X_{i16} - 285.116X_{i17} - 44.619X_{i18} - 17.826X_{i20} + 7.342X_{i21} + 15.237X_{i23} + 146.327X_{i26} + 30.317X_{i28} + 114.095X_{i29}$$

The final model for output V10 is:

$$E\{Y_i\} = 226.799 + 3.524X_{i4} + 126.555X_{i5} + 13.708X_{i7} + 12.726X_{i23}$$

III Conclusion

i. Outcome Interpretation

Base on our previous analysis and the models that we have built, we can find that the actual sales prices are highly positively related to the price of the unit at the beginning of the project. What’s more, the land price index for the base year negatively influenced the actual sales prices. The CPI of housing, water, fuel & power in the base year is also significantly influencing the actual sales prices. Other factors such as Total preliminary estimated construction cost based on the prices at the beginning of the project are also important, but the significance is not as important as the three factors that we mentioned before. Factors like lot project or total floor area of the building are not relevant to actual sales prices.

As for the second factors that we have predicted (actual construction costs), we find out that one variable is significant with our outcome, that is preliminary estimated construction cost based on the prices at the beginning of the project. Other factors such as duration of construction, total preliminary estimated construction cost based on the prices at the beginning of the project are also very important. According to the stepwise selection and observation of the connection between different observations, we believe this is reasonable and totally make sense.

ii. Improvement

- Although our outcome is extremely great, R2 is pretty high, we still have some improvement space. For example, our date observations are only 372, which could be higher if we want a more accurate result.
- The data that we have received is contained 5 different lags, but we only used one of them, because this contains some time series problems that we cannot resolve based on our current knowledge. We believe that if we can all of the data and compared the results in different periods, our results might be better and more convincing.
- All the price is time sensitive data, and we can collect more data at different times to improve our prediction. We can still utilize feature engineering in order to create more significant variables to prove our results.

IV. Reference:

Kutner, M. H. (Ed.). (2005). Applied linear statistical models (5th ed). Boston: McGraw-Hill Irwin.

Hongmei, Jiang. (2019) . Chapter3_Graphs_BrownTest. [Source code]. https://canvas.northwestern.edu/courses/89947/files/folder/R%20Codes?preview=6235941

Hongmei, Jiang. (2019) . Chapter10_Section6_SurgicalUnit. [Source code]. https://canvas.northwestern.edu/courses/89947/files/folder/R%20Codes?preview=6558400

Hongmei, Jiang. (2019) . Chapter12_glmnet. [Source code]. https://canvas.northwestern.edu/courses/89947/files/folder/R%20Codes?preview=6235967

V. Appendix:

R code for predicting V-9

R code for predicting V-10

STAT 350 Final Project of predicting V-9

Xiaotian Ding, Jie Gu, De Lu, Huaiyi Liu

2019/3/10

```
# Import the data and pretreatment
load("~/Desktop/Study in NU/Winter/Regression analysis/Final/train & test.RData")
Zipcode= as.factor(train$`V-1`)
Zipcodetest=as.factor(test$`V-1`)
#strandized for train
y9=as.matrix(train$`V-9`)
colnames(y9)=c("V-9")
y10=as.matrix(train$`V-10`)
colnames(y10)=c("V-10")
train.v9<- cbind(Zipcode,train[2:27],y9)
train.v10<- cbind(Zipcode,train[2:27],y10)
for(i in 2:27)
 {
  train.v9[,i] <-(train[,i]-mean(train[,i])) /sd(train[,i])
}
for(i in 2:27)
 {
  train.v10[,i] <-(train[,i]-mean(train[,i])) /sd(train[,i])
}

#strandized for test data
yt9=as.matrix(test$`V-9`)
colnames(yt9)=c("V-9")
yt10=as.matrix(test$`V-10`)
colnames(yt10)=c("V-10")
test.v9<- cbind(test[1:27],yt9)
test.v10<- cbind(test[1:27],yt10)
for(i in 2:27)
 {
  test.v9[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
for(i in 2:27)
 {
  test.v10[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
```

```
# scatter plot & cor
fit<- lm(train$`V-9`~ ., data = train.v9)
summary(fit)
```

```
##
## Call:
## lm(formula = train$`V-9` ~ ., data = train.v9)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -866.96  -57.30   -3.45   48.17  694.97
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 1379.301     43.027  32.057  < 2e-16 ***
## Zipcode2      29.064     49.548   0.587 0.557944
## Zipcode3      25.886     43.188   0.599 0.549383
## Zipcode4       7.861     52.036   0.151 0.880033
## Zipcode5     -39.912     55.242  -0.722 0.470579
## Zipcode6      41.466     52.058   0.797 0.426375
## Zipcode7     -14.318     54.736  -0.262 0.793830
## Zipcode8      15.753     55.631   0.283 0.777249
## Zipcode9     -31.679    101.782  -0.311 0.755843
## Zipcode10    -73.789     86.248  -0.856 0.392964
## Zipcode11    -25.086     87.989  -0.285 0.775766
## Zipcode12    -47.802     65.352  -0.731 0.465094
## Zipcode13      8.717     67.690   0.129 0.897624
## Zipcode14    -39.447     59.803  -0.660 0.510024
## Zipcode15   -113.089     74.395  -1.520 0.129581
## Zipcode16    -40.856     78.102  -0.523 0.601296
## Zipcode17    -91.548     63.516  -1.441 0.150579
## Zipcode18    -66.498     67.816  -0.981 0.327633
## Zipcode19    -78.702     66.889  -1.177 0.240325
## Zipcode20    -24.287     60.449  -0.402 0.688155
## `V-2`         70.603     40.080   1.762 0.079209 .
## `V-3`        -62.621     31.587  -1.982 0.048375 *
## `V-4`         12.618     19.827   0.636 0.525003
## `V-5`         -4.853     38.828  -0.125 0.900624
## `V-6`         -5.382     17.112  -0.314 0.753376
## `V-7`         95.723      8.985  10.654  < 2e-16 ***
## `V-8`       1115.815     22.388  49.839  < 2e-16 ***
## `V-11`       -29.914     20.610  -1.451 0.147764
## `V-12`       127.711    219.259   0.582 0.560710
## `V-13`        19.304    153.173   0.126 0.899800
## `V-14`        31.474     20.272   1.553 0.121627
## `V-15`       778.435    150.325   5.178 4.21e-07 ***
## `V-16`        80.756     45.294   1.783 0.075646 .
## `V-17`      -917.386     82.097 -11.174  < 2e-16 ***
## `V-18`        19.316     34.794   0.555 0.579216
## `V-19`      -236.551     39.067  -6.055 4.37e-09 ***
## `V-20`        77.938     20.509   3.800 0.000176 ***
## `V-21`       126.122     87.688   1.438 0.151435
## `V-22`      -168.009    110.769  -1.517 0.130424
## `V-23`        64.753     39.041   1.659 0.098288 .
## `V-24`       -75.117     47.310  -1.588 0.113436
## `V-25`       332.551    414.480   0.802 0.423022
## `V-26`       210.348    332.101   0.633 0.526984
## `V-27`      -108.968     38.840  -2.806 0.005365 **
## `V-28`        32.413     24.618   1.317 0.189009
## `V-29`      -307.642     74.979  -4.103 5.31e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 145.8 on 288 degrees of freedom
## Multiple R-squared:  0.9862, Adjusted R-squared:  0.9841
## F-statistic:   458 on 45 and 288 DF,  p-value: < 2.2e-16
```

```r
cor(train.v9[,c(2:28)])
```

```
##                 V-2          V-3         V-4         V-5         V-6
## V-2   1.000000000  0.945600958  0.77436762  0.23420508  0.21230859
## V-3   0.945600958  1.000000000  0.64019309  0.15528248  0.12979894
## V-4   0.774367618  0.640193087  1.00000000  0.57121240  0.33816723
## V-5   0.234205081  0.155282482  0.57121240  1.00000000  0.32722179
## V-6   0.212308594  0.129798937  0.33816723  0.32722179  1.00000000
## V-7   0.161547288  0.089225497  0.18086665  0.06782641  0.14136645
## V-8   0.226480811  0.145503571  0.47418223  0.80942131  0.16406550
## V-11 -0.035418005 -0.013619707  0.02290495  0.23425775 -0.06469386
## V-12  0.089149149  0.082405943  0.32119612  0.79164543 -0.19515942
## V-13  0.088261381  0.076359505  0.31999660  0.79468122 -0.18162136
## V-14 -0.031042870 -0.007498359  0.05373985  0.24842339 -0.06594751
## V-15  0.079502198  0.075962603  0.30494320  0.77237472 -0.20802750
## V-16  0.092661957  0.094008848  0.29144067  0.70758556 -0.17878957
## V-17  0.092216627  0.086045671  0.31811051  0.78139809 -0.18702009
## V-18  0.067722258  0.051402223  0.20398832  0.48986363 -0.05370173
## V-19  0.063634669  0.061394585  0.25013127  0.64261538 -0.16845274
## V-20 -0.007826318 -0.030715653 -0.04911039 -0.21740676  0.11616290
## V-21  0.083525605  0.073809085  0.31314787  0.78345004 -0.20180050
## V-22  0.100604965  0.092276694  0.32827391  0.78788394 -0.19693094
## V-23  0.094428570  0.062129772  0.31781905  0.72055691 -0.16593281
## V-24  0.082614838  0.062240681  0.26826007  0.64481091 -0.08200588
## V-25  0.089286179  0.073363554  0.32671357  0.79833830 -0.18004635
## V-26  0.090209687  0.074963521  0.32844952  0.79873355 -0.18638055
## V-27  0.098501142  0.066699364  0.32827781  0.72016802 -0.14959341
## V-28  0.081782289  0.065819345  0.15407731  0.30938116 -0.01506138
## V-29  0.079099698  0.074670140  0.30292846  0.78219642 -0.19173899
## V-9   0.248620088  0.151985665  0.49638489  0.79090924  0.18391391
##                 V-7         V-8         V-11        V-12         V-13
## V-2   0.161547288  0.22648081 -0.035418005  0.08914915  0.088261381
## V-3   0.089225497  0.14550357 -0.013619707  0.08240594  0.076359505
## V-4   0.180866645  0.47418223  0.022904947  0.32119612  0.319996595
## V-5   0.067826410  0.80942131  0.234257753  0.79164543  0.794681216
## V-6   0.141366451  0.16406550 -0.064693859 -0.19515942 -0.181621357
## V-7   1.000000000  0.01776631  0.002002689 -0.01102056  0.001928748
## V-8   0.017766307  1.00000000  0.214791473  0.62970430  0.634816454
## V-11  0.002002689  0.21479147  1.000000000  0.31244811  0.344567028
## V-12 -0.011020564  0.62970430  0.312448115  1.00000000  0.990115542
## V-13  0.001928748  0.63481645  0.344567028  0.99011554  1.000000000
## V-14 -0.019721226  0.24596566  0.860365531  0.31564739  0.340203213
## V-15 -0.024051443  0.61691478  0.332852738  0.98696187  0.965649585
## V-16 -0.001528973  0.54631418  0.413873083  0.91047002  0.911852344
## V-17 -0.007634694  0.61852139  0.360611448  0.98089806  0.976602879
## V-18  0.103500897  0.41133991  0.234032892  0.51878542  0.526198919
## V-19 -0.006441351  0.54873449  0.299067350  0.80590755  0.775279122
## V-20  0.074979304 -0.13666349 -0.264273120 -0.35712077 -0.289603798
## V-21 -0.001781932  0.61813693  0.327402884  0.99252339  0.983217541
## V-22  0.005911640  0.61471306  0.307851986  0.99291719  0.980110935
## V-23  0.046973086  0.56744724  0.111823359  0.85524646  0.868914803
## V-24  0.066728819  0.51811642  0.409866559  0.74751228  0.810329992
## V-25  0.017701519  0.63912703  0.346748048  0.98283160  0.993948015
## V-26  0.012712520  0.64038848  0.328935009  0.98773368  0.992846711
```

```
## V-27  0.041148951  0.58009504  0.101545061  0.83897307  0.859104077
## V-28  0.097650836  0.27813186  0.104124516  0.29277178  0.286868723
## V-29 -0.020623125  0.63038431  0.302106965  0.98201195  0.970009704
## V-9   0.110731006  0.97801992  0.176425972  0.59629978  0.604221911
##               V-14        V-15         V-16         V-17        V-18
## V-2  -0.031042870  0.07950220  0.092661957  0.092216627  0.06772226
## V-3  -0.007498359  0.07596260  0.094008848  0.086045671  0.05140222
## V-4   0.053739850  0.30494320  0.291440666  0.318110514  0.20398832
## V-5   0.248423394  0.77237472  0.707585561  0.781398086  0.48986363
## V-6  -0.065947512 -0.20802750 -0.178789572 -0.187020090 -0.05370173
## V-7  -0.019721226 -0.02405144 -0.001528973 -0.007634694  0.10350090
## V-8   0.245965661  0.61691478  0.546314176  0.618521388  0.41133991
## V-11  0.860365531  0.33285274  0.413873083  0.360611448  0.23403289
## V-12  0.315647390  0.98696187  0.910470016  0.980898057  0.51878542
## V-13  0.340203213  0.96564959  0.911852344  0.976602879  0.52619892
## V-14  1.000000000  0.32141733  0.429650641  0.377683833  0.22975516
## V-15  0.321417327  1.00000000  0.891777513  0.965041224  0.53798776
## V-16  0.429650641  0.89177751  1.000000000  0.945888259  0.47463555
## V-17  0.377683833  0.96504122  0.945888259  1.000000000  0.51550667
## V-18  0.229755159  0.53798776  0.474635548  0.515506674  1.00000000
## V-19  0.280428299  0.85592540  0.710762676  0.763396891  0.76005321
## V-20 -0.223180633 -0.42477138 -0.460695728 -0.421048499 -0.04807152
## V-21  0.314238352  0.98479930  0.905593128  0.970529084  0.52651551
## V-22  0.308650562  0.98188751  0.922203466  0.977496291  0.54096321
## V-23  0.094598910  0.83560127  0.752719249  0.849345735  0.47191261
## V-24  0.385324013  0.67979203  0.660607493  0.721302456  0.51287317
## V-25  0.337258655  0.95920948  0.886957004  0.963501199  0.55244157
## V-26  0.321973307  0.96848328  0.892081494  0.969094040  0.54969275
## V-27  0.139606014  0.79419440  0.681317005  0.818461854  0.39502159
## V-28  0.169608370  0.30630060  0.229200249  0.291059956  0.86463576
## V-29  0.299937377  0.98315243  0.866570249  0.946223759  0.53018537
## V-9   0.193866852  0.58392013  0.487077618  0.564760691  0.39517449
##               V-19         V-20         V-21        V-22        V-23
## V-2   0.063634669 -0.007826318  0.083525605  0.10060497  0.09442857
## V-3   0.061394585 -0.030715653  0.073809085  0.09227669  0.06212977
## V-4   0.250131268 -0.049110392  0.313147874  0.32827391  0.31781905
## V-5   0.642615376 -0.217406759  0.783450035  0.78788394  0.72055691
## V-6  -0.168452740  0.116162896 -0.201800496 -0.19693094 -0.16593281
## V-7  -0.006441351  0.074979304 -0.001781932  0.00591164  0.04697309
## V-8   0.548734491 -0.136663489  0.618136933  0.61471306  0.56744724
## V-11  0.299067350 -0.264273120  0.327402884  0.30785199  0.11182336
## V-12  0.805907545 -0.357120769  0.992523387  0.99291719  0.85524646
## V-13  0.775279122 -0.289603798  0.983217541  0.98011094  0.86891480
## V-14  0.280428299 -0.223180633  0.314238352  0.30865056  0.09459891
## V-15  0.855925405 -0.424771384  0.984799301  0.98188751  0.83560127
## V-16  0.710762676 -0.460695728  0.905593128  0.92220347  0.75271925
## V-17  0.763396891 -0.421048499  0.970529084  0.97749629  0.84934574
## V-18  0.760053214 -0.048071518  0.526515511  0.54096321  0.47191261
## V-19  1.000000000 -0.277789904  0.817092309  0.81052043  0.63333981
## V-20 -0.277789904  1.000000000 -0.359643069 -0.36626055 -0.22614493
## V-21  0.817092309 -0.359643069  1.000000000  0.99053656  0.84433800
## V-22  0.810520432 -0.366260547  0.990536564  1.00000000  0.84869783
## V-23  0.633339813 -0.226144932  0.844337998  0.84869783  1.00000000
## V-24  0.531494188  0.097835932  0.745168073  0.73102304  0.65975292
```

```
## V-25  0.781298186 -0.248323465  0.977863958  0.97321585  0.88823633
## V-26  0.792637630 -0.283334553  0.981947612  0.97801582  0.89700001
## V-27  0.570548168 -0.069144581  0.824139284  0.81998426  0.91703848
## V-28  0.561385894  0.037568332  0.296720859  0.31415987  0.25026572
## V-29  0.828660567 -0.330418745  0.975786332  0.97155143  0.83140014
## V-9   0.523059207 -0.061325422  0.589593428  0.58133547  0.56541923
##             V-24        V-25        V-26        V-27        V-28
## V-2   0.08261484  0.08928618  0.09020969  0.09850114  0.08178229
## V-3   0.06224068  0.07336355  0.07496352  0.06669936  0.06581934
## V-4   0.26826007  0.32671357  0.32844952  0.32827781  0.15407731
## V-5   0.64481091  0.79833830  0.79873355  0.72016802  0.30938116
## V-6  -0.08200588 -0.18004635 -0.18638055 -0.14959341 -0.01506138
## V-7   0.06672882  0.01770152  0.01271252  0.04114895  0.09765084
## V-8   0.51811642  0.63912703  0.64038848  0.58009504  0.27813186
## V-11  0.40986656  0.34674805  0.32893501  0.10154506  0.10412452
## V-12  0.74751228  0.98283160  0.98773368  0.83897307  0.29277178
## V-13  0.81032999  0.99394802  0.99284671  0.85910408  0.28686872
## V-14  0.38532401  0.33725865  0.32197331  0.13960601  0.16960837
## V-15  0.67979203  0.95920948  0.96848328  0.79419440  0.30630060
## V-16  0.66060749  0.88695700  0.89208149  0.68131701  0.22920025
## V-17  0.72130246  0.96350120  0.96909404  0.81846185  0.29105996
## V-18  0.51287317  0.55244157  0.54969275  0.39502159  0.86463576
## V-19  0.53149419  0.78129819  0.79263763  0.57054817  0.56138589
## V-20  0.09783593 -0.24832347 -0.28333455 -0.06914458  0.03756833
## V-21  0.74516807  0.97786396  0.98194761  0.82413928  0.29672086
## V-22  0.73102304  0.97321585  0.97801582  0.81998426  0.31415987
## V-23  0.65975292  0.88823633  0.89700001  0.91703848  0.25026572
## V-24  1.00000000  0.83419437  0.80279114  0.70242848  0.25925649
## V-25  0.83419437  1.00000000  0.99785756  0.87754156  0.30716315
## V-26  0.80279114  0.99785756  1.00000000  0.88162945  0.31251731
## V-27  0.70242848  0.87754156  0.88162945  1.00000000  0.25408508
## V-28  0.25925649  0.30716315  0.31251731  0.25408508  1.00000000
## V-29  0.73163217  0.96492985  0.96786639  0.80227328  0.29412577
## V-9   0.51839547  0.61808272  0.61802696  0.57921009  0.26797608
##             V-29         V-9
## V-2   0.07909970  0.24862009
## V-3   0.07467014  0.15198567
## V-4   0.30292846  0.49638489
## V-5   0.78219642  0.79090924
## V-6  -0.19173899  0.18391391
## V-7  -0.02062312  0.11073101
## V-8   0.63038431  0.97801992
## V-11  0.30210697  0.17642597
## V-12  0.98201195  0.59629978
## V-13  0.97000970  0.60422191
## V-14  0.29993738  0.19386685
## V-15  0.98315243  0.58392013
## V-16  0.86657025  0.48707762
## V-17  0.94622376  0.56476069
## V-18  0.53018537  0.39517449
## V-19  0.82866057  0.52305921
## V-20 -0.33041874 -0.06132542
## V-21  0.97578633  0.58959343
## V-22  0.97155143  0.58133547
```

```
## V-23  0.83140014  0.56541923
## V-24  0.73163217  0.51839547
## V-25  0.96492985  0.61808272
## V-26  0.96786639  0.61802696
## V-27  0.80227328  0.57921009
## V-28  0.29412577  0.26797608
## V-29  1.00000000  0.60376480
## V-9   0.60376480  1.00000000
```

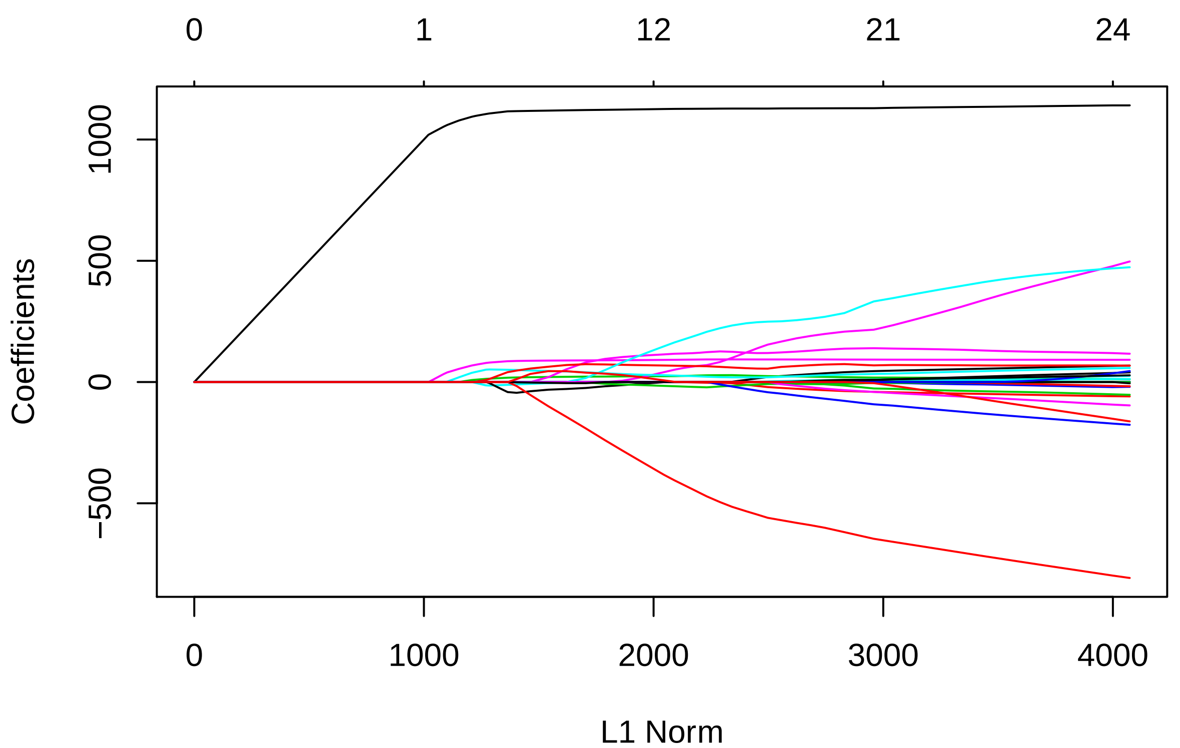
```
# Lasso to determine variable
library("glmnet")
```

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

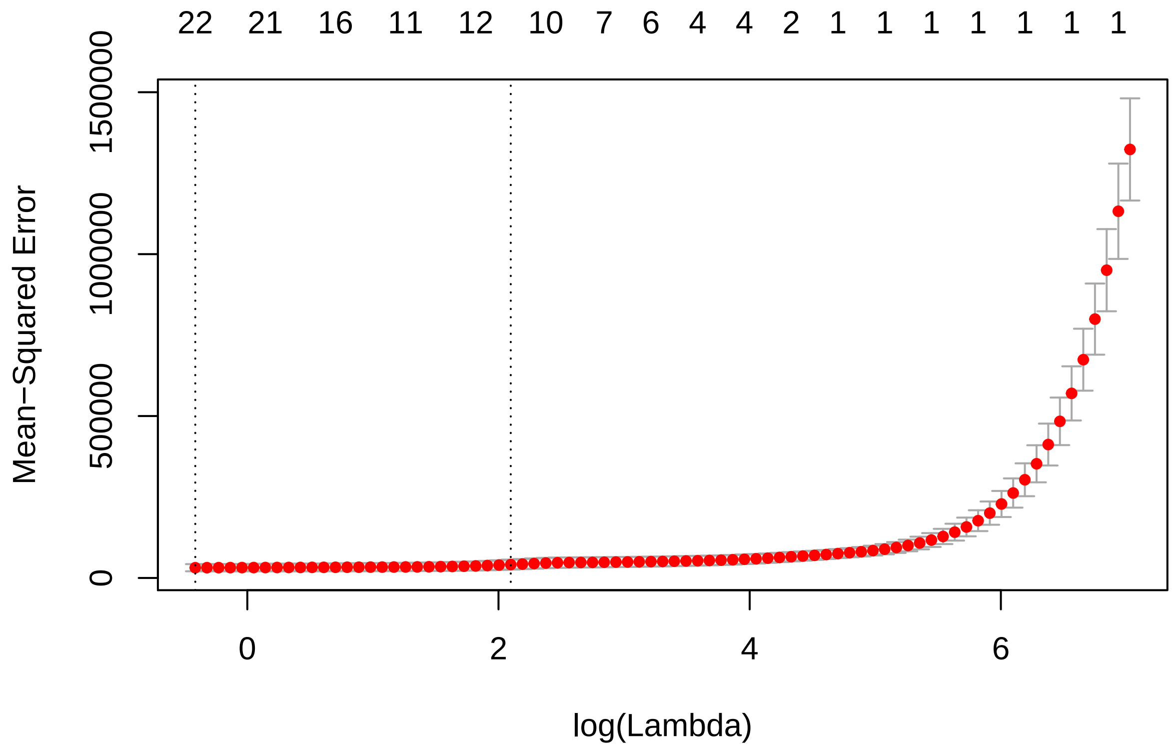
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
x.simple=as.matrix(train.v9[,2:27])
y=train.v9$`V-9`
fitlasso=glmnet(x.simple,y,alpha = 1)
plot(fitlasso)
```




```
cv.lasso=cv.glmnet(x.simple,y)
plot(cv.lasso)
```




```r
coef(cv.lasso)
```

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 1359.595808
## V-2            .
## V-3            .
## V-4           22.277914
## V-5            .
## V-6            .
## V-7           89.183305
## V-8         1121.712305
## V-11           .
## V-12           .
## V-13           .
## V-14           .
## V-15           .
## V-16         -24.459751
## V-17        -191.188370
## V-18          -1.247671
## V-19           .
## V-20          40.308285
## V-21          80.944313
## V-22           .
## V-23          73.542214
## V-24           .
## V-25           .
## V-26          16.447744
## V-27           .
## V-28          -4.625408
## V-29          38.730674
```

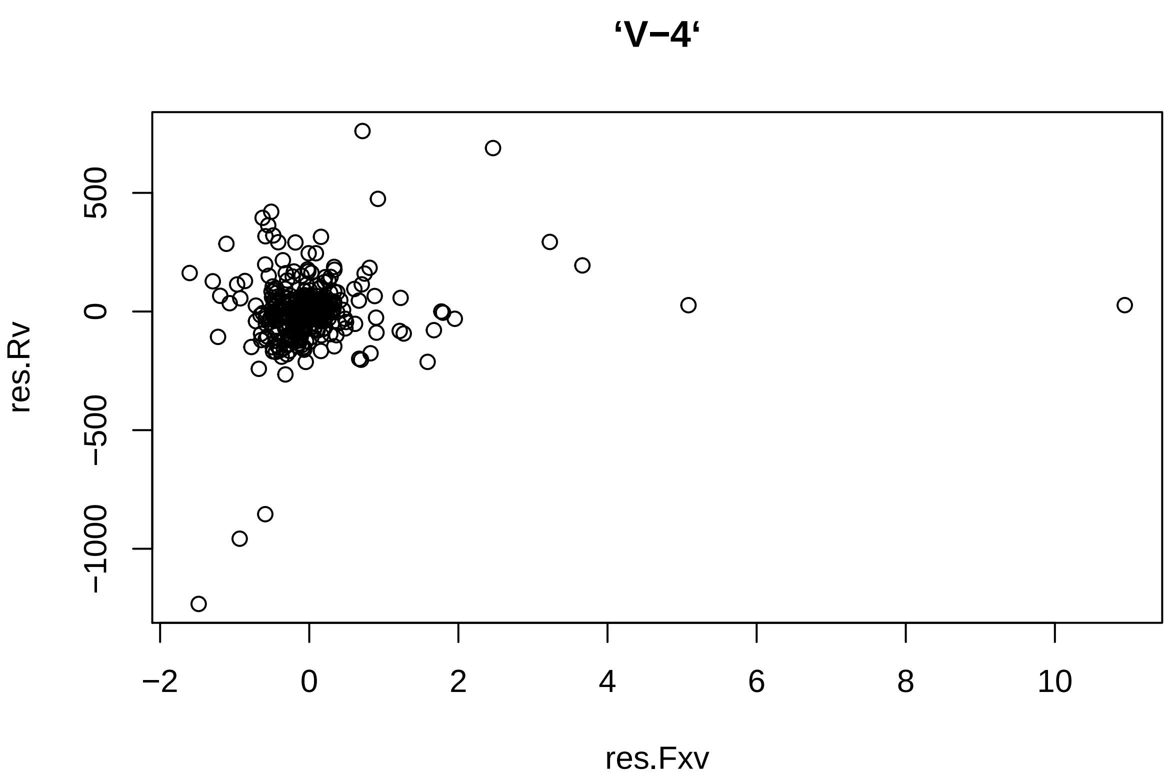
```r
# added variable factor to determine ^
datamodel1=data.frame(train.v9[,c(4,7,8,14,15,16,18,19,21,24,26,27,28)])
# Model 1
fitmodel1=lm(datamodel1$V.9~.,data = datamodel1)
summary(fitmodel1)$r.squared
```

```
## [1] 0.9825749
```

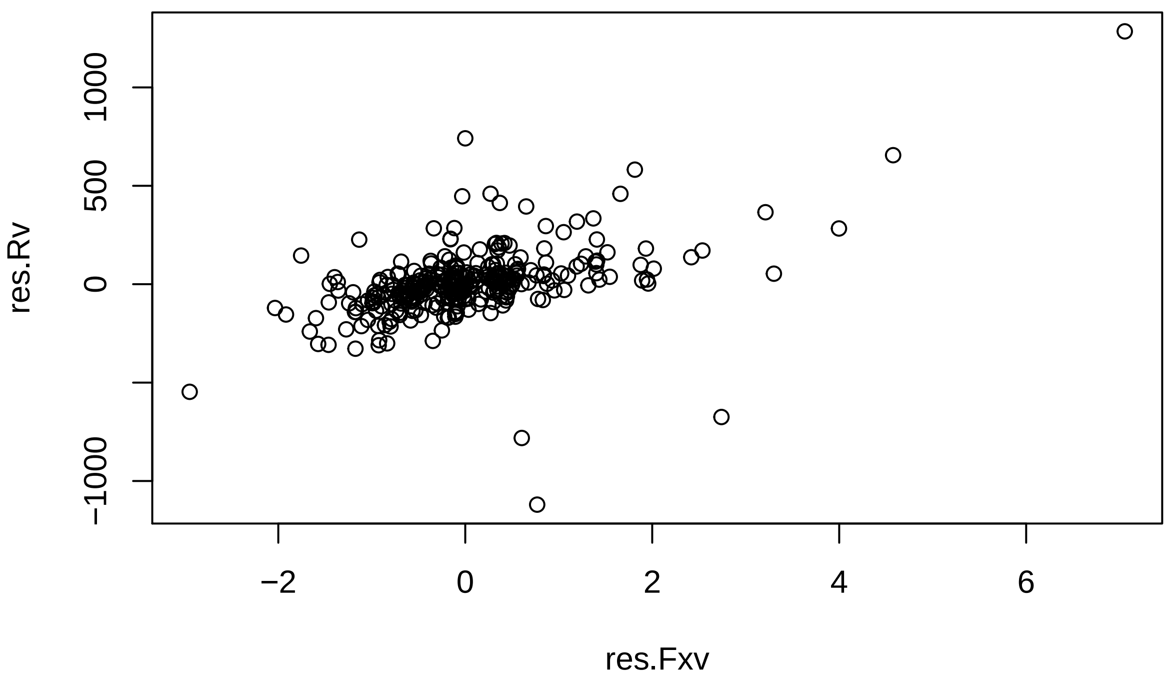
```r
summary(fitmodel1)$adj.r.squared
```

```
## [1] 0.9819235
```

```r
colData <- list("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
names(colData) <- c("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
removeXList <- colData

for (rmX in removeXList){
  tmpV <- colData
  tmpV[[rmX]] = NULL
  test.Rv=lm(as.formula(paste("`V-9` ~", paste(tmpV, collapse = "+"))), data = train.v9)
  res.Rv= test.Rv$residuals

  test.Fxv=lm(as.formula(paste(paste(rmX," ~"), paste(tmpV, collapse = "+"))), data = train.v9)
  res.Fxv= test.Fxv$residuals

  plot(res.Fxv,res.Rv,main = rmX)
}
```

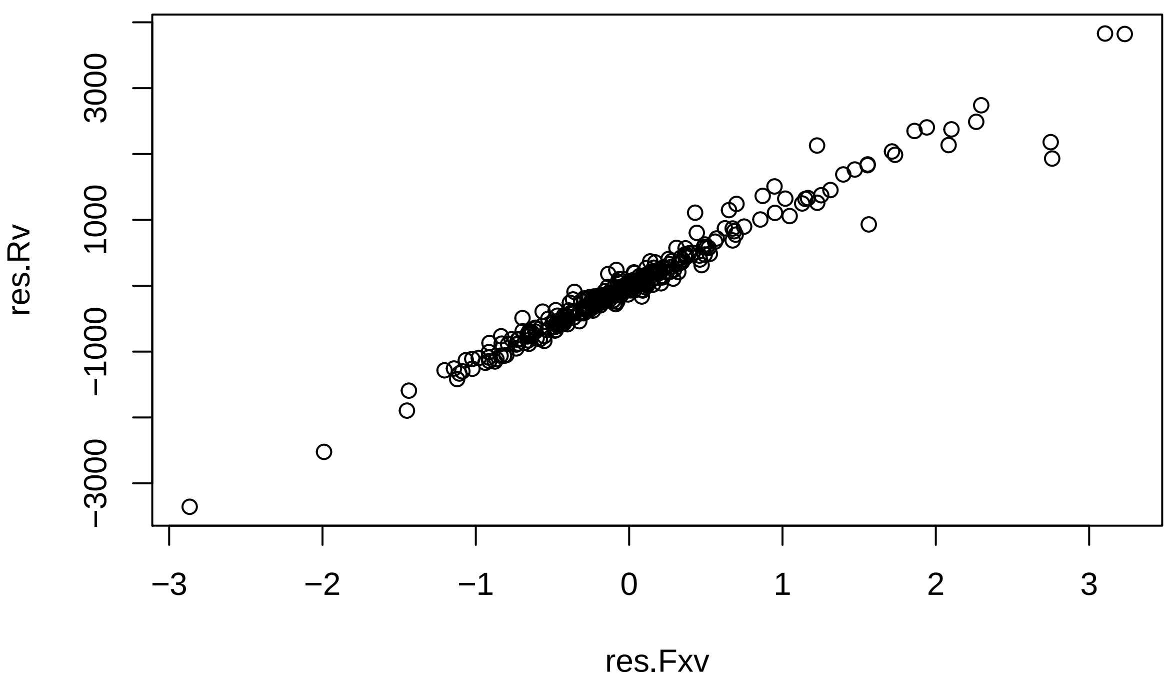


‘V−7‘



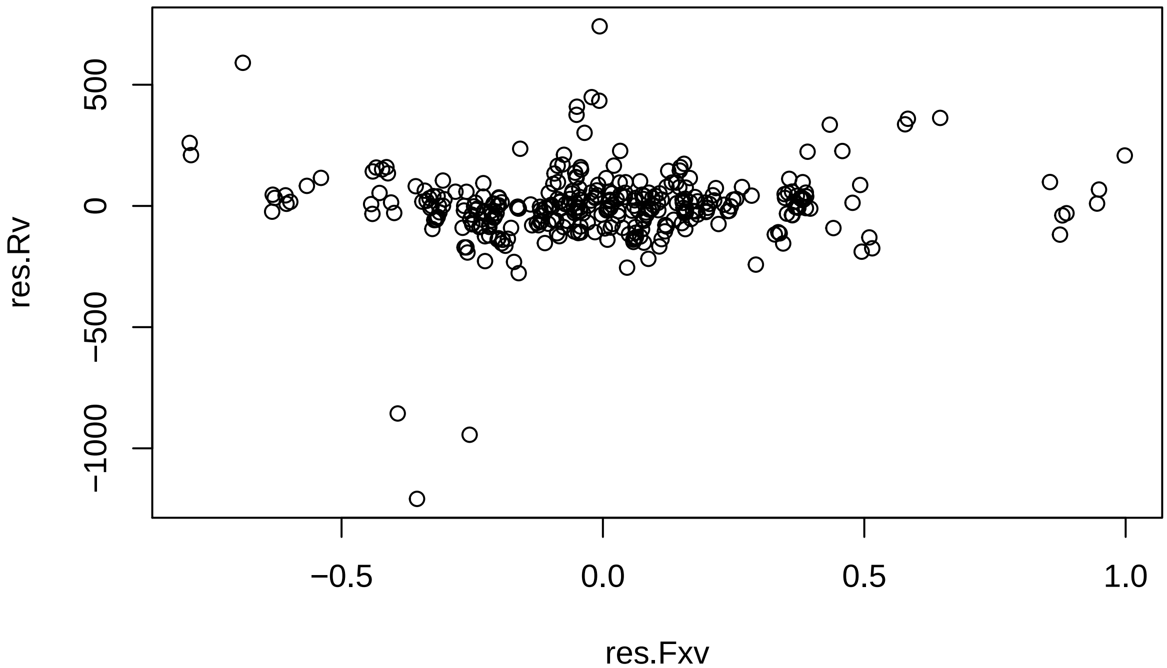
res.Rv
1000
500
0
−1000
−2
0
2
4
6
res.Fxv

‘V−8‘

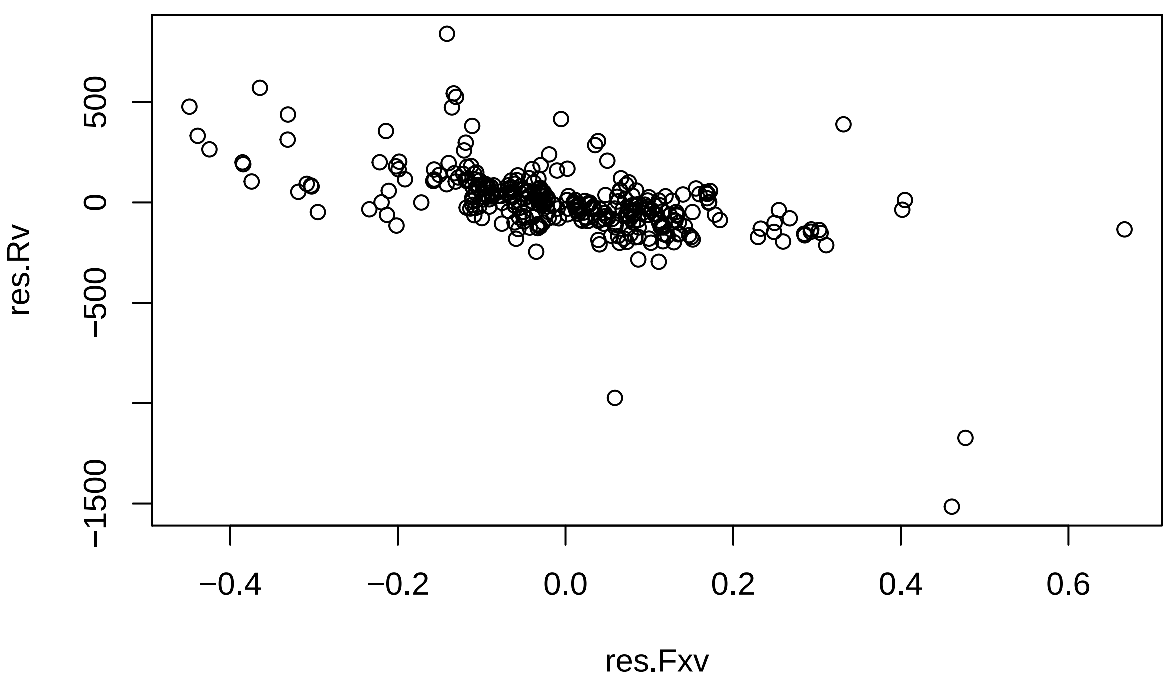


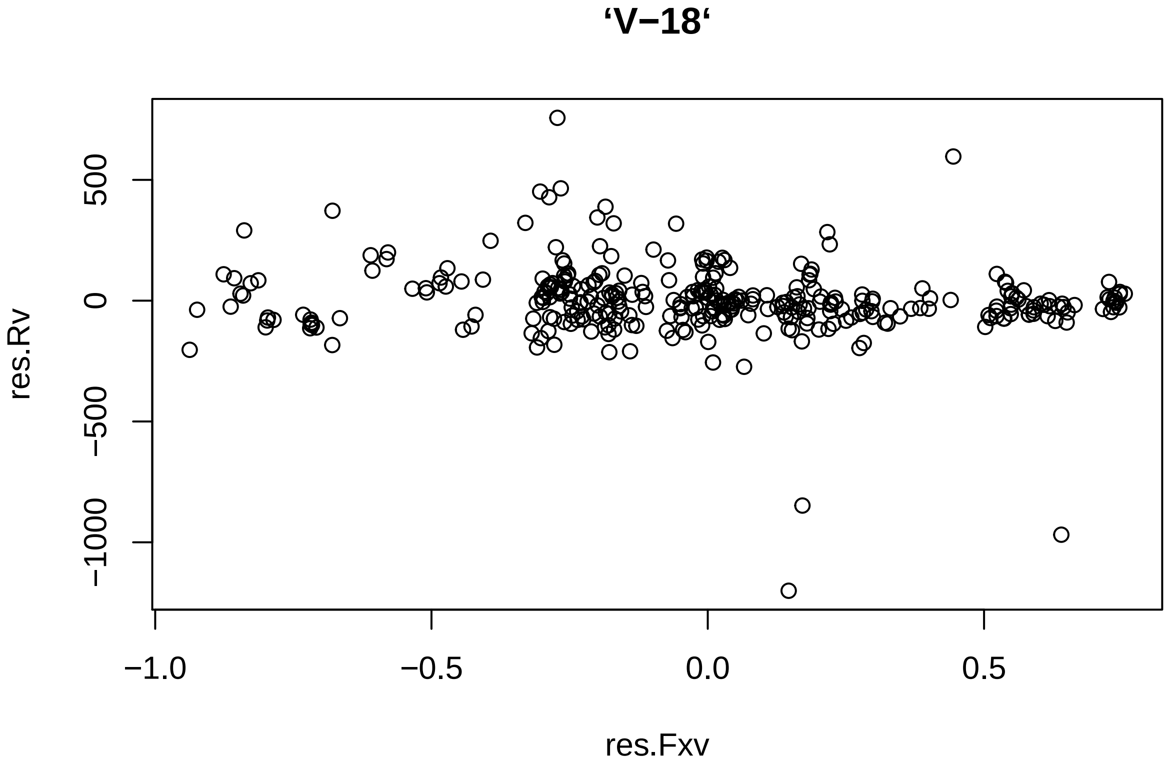
res.Rv
3000
1000
−1000
−3000
−3
−2
−1
0
1
2
3
res.Fxv

‘V−16‘

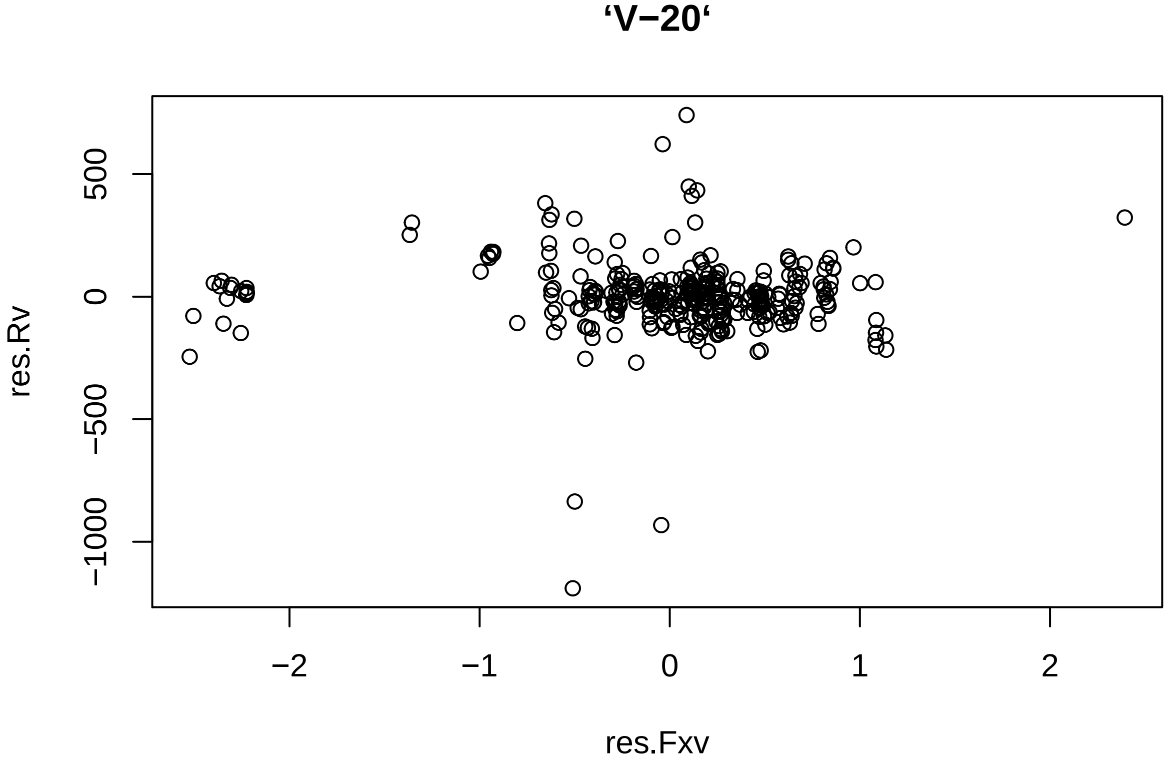

res.Rv
500
0
−500
−1000
−0.5
0.0
0.5
1.0
res.Fxv

‘V−17‘

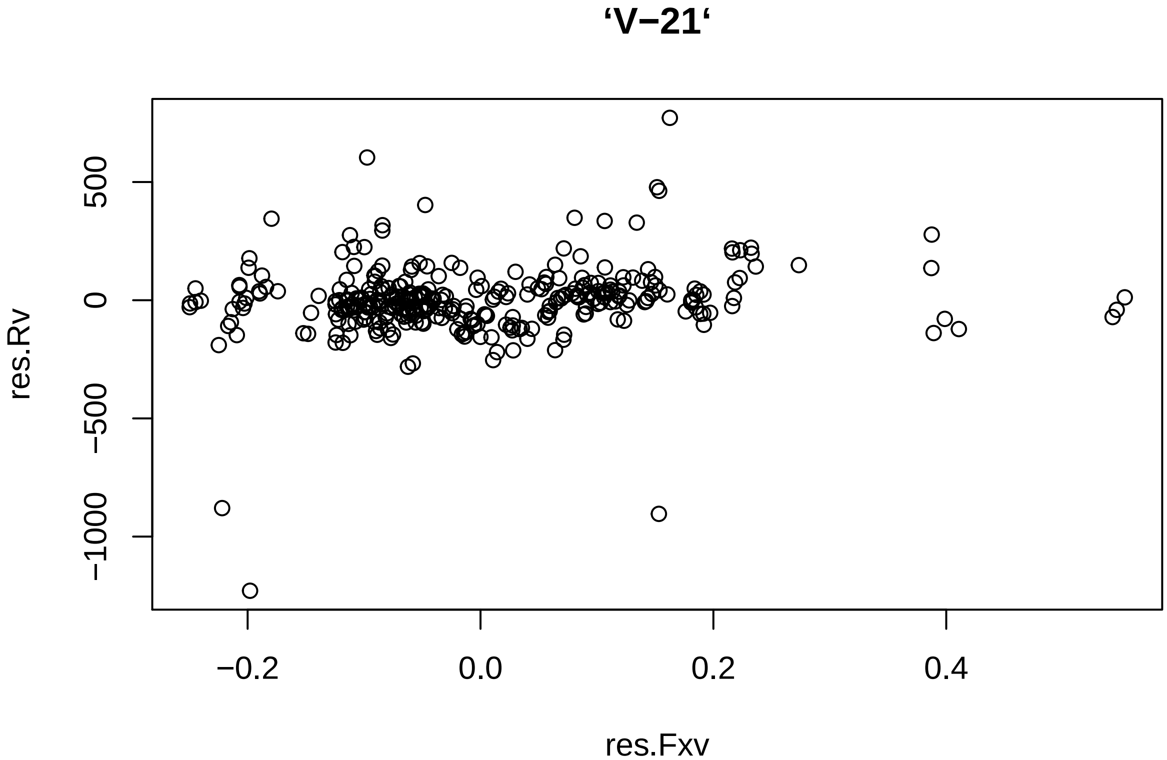

res.Rv
500
0
−500
−1500
−0.4
−0.2
0.0
0.2
0.4
0.6
res.Fxv

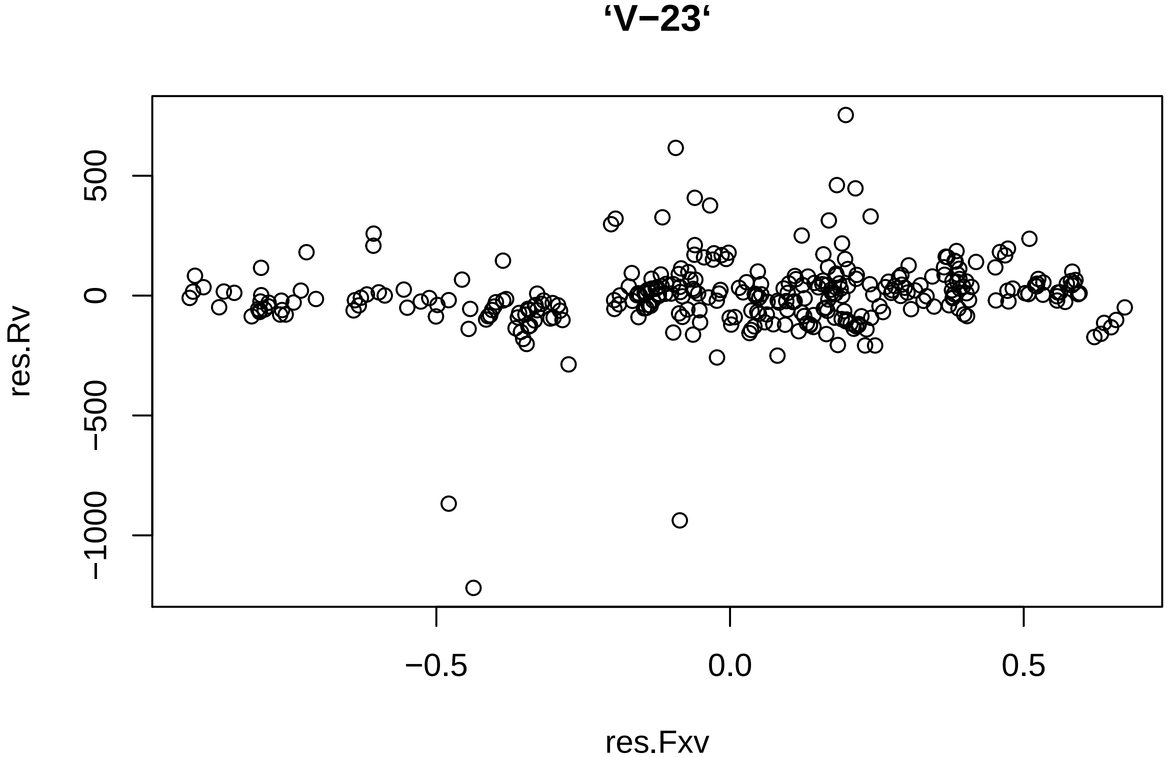


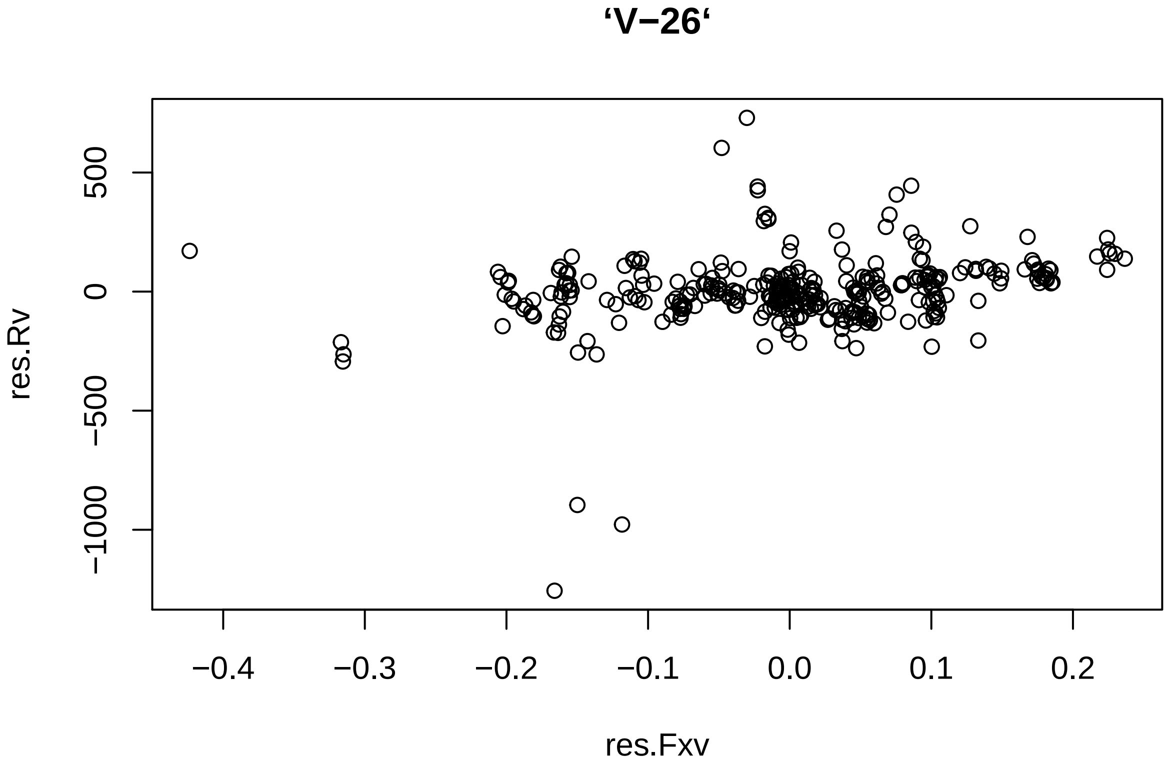
‘V−18‘
500
0
−500
−1000
res.Rv
−1.0
−0.5
0.0
0.5
res.Fxv

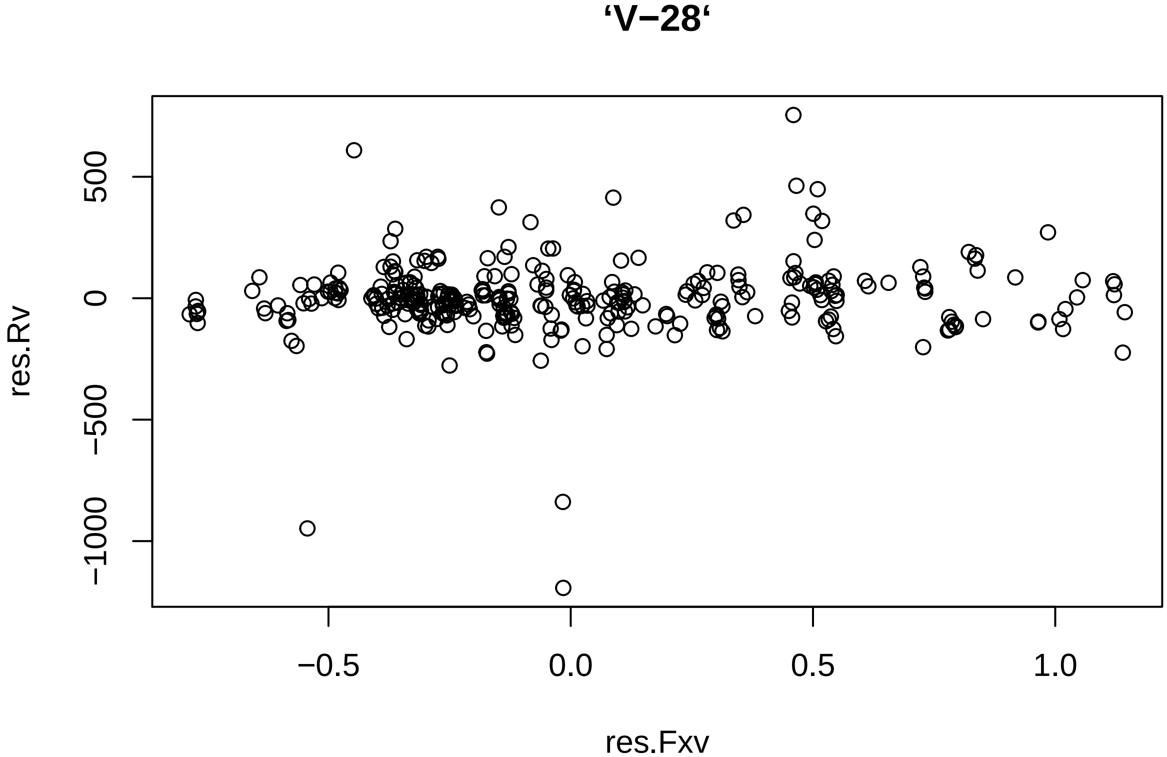


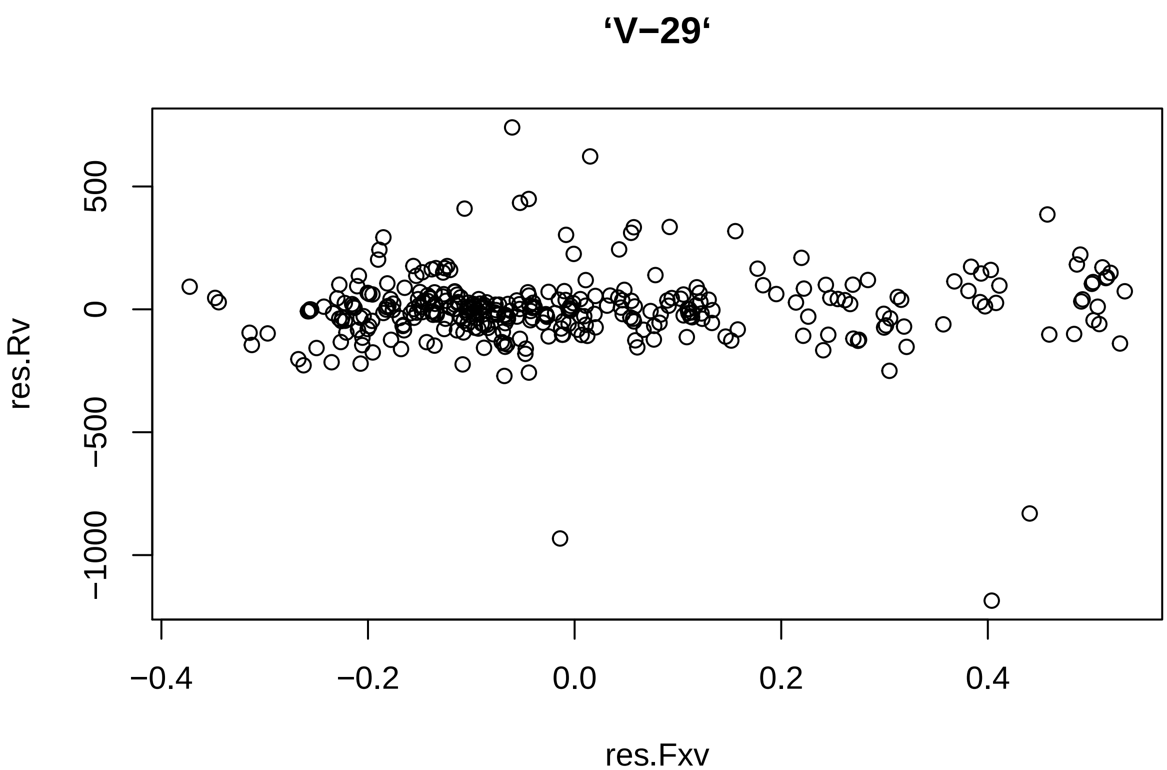
‘V−20‘
500
0
−500
−1000
res.Rv
−2
−1
0
1
2
res.Fxv


‘V−21‘
500
0
−500
−1000
res.Rv
−0.2
0.0
0.2
0.4
res.Fxv


‘V−23‘
500
0
−500
−1000
res.Rv
−0.5
0.0
0.5
res.Fxv


‘V−26‘
res.Rv
500
0
−500
−1000
−0.4
−0.3
−0.2
−0.1
0.0
0.1
0.2
res.Fxv


‘V−28‘
res.Rv
500
0
−500
−1000
−0.5
0.0
0.5
1.0
res.Fxv




```
#Brown test whether constant variance and transformation for Model 1
resmodel1=fitmodel1$residuals
mmodel1=mean(datamodel1$V.9)
nmodel1=dim(datamodel1)[1]
p1=13
#1. Break the residuals into two groups.
Group1 <- resmodel1[datamodel1$V.9<mmodel1]
Group2 <-resmodel1[datamodel1$V.9>=mmodel1]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel1-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
 t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
 t
```

```
## [1] -5.811487
```

```
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)   # find the catical value
```

```
## [1] 1.967405
```

```r
# Weighted tranformation for model 1
wts <- 1/fitted(lm(abs(residuals(fitmodel1)) ~ ., data = datamodel1))^2

fitmodel1weight <- lm(datamodel1$V.9~ .,data = datamodel1, weights=wts)
datamodel1weight=cbind(datamodel1[1:12],datamodel1$V.9*wts)
summary(fitmodel1weight)$r.squared
```

```
## [1] 0.9997356
```

```r
summary(fitmodel1weight)$adj.r.squared
```

```
## [1] 0.9997257
```

```r
#Brown test whether constant variance and transformation for Model 1 after tranformation
resmodel1b=fitmodel1weight $residuals
mmodel1=mean(datamodel1weight$`datamodel1$V.9 * wts`)
nmodel1=dim(datamodel1weight)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel1b[datamodel1weight$`datamodel1$V.9 * wts`<mmodel1]
Group2 <-resmodel1b[datamodel1weight$`datamodel1$V.9 * wts`>=mmodel1]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel1-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
t
```

```
## [1] 1.434477
```

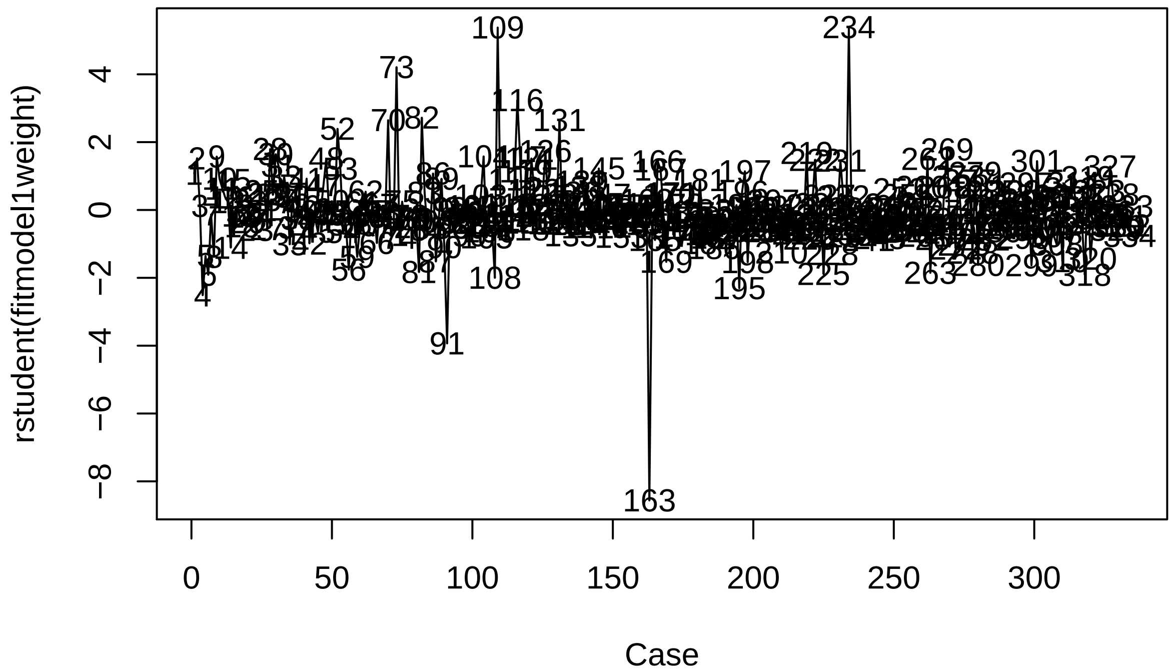
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)   # find the catical value
```

```
## [1] 1.967405
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel1-p1-1))
```

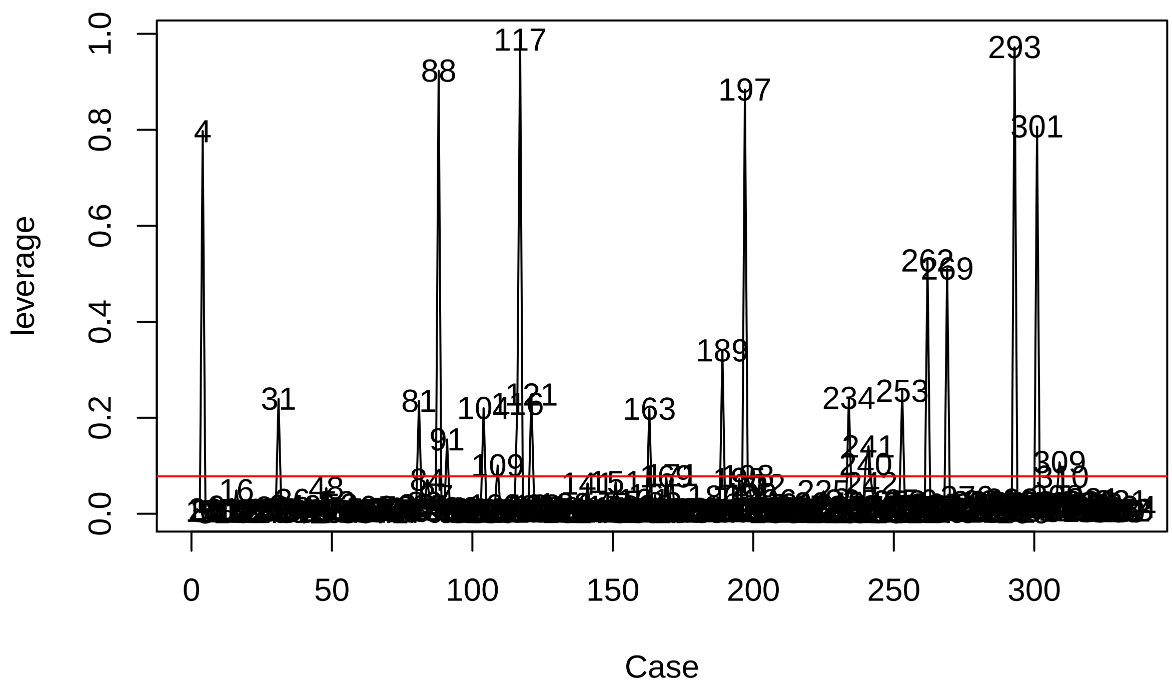
```
## [1] 0.1524126
```

```r
#y outlier for model1
Case <- c(1:nmodel1)
plot(Case, rstudent(fitmodel1weight), type="l")
text(Case, rstudent(fitmodel1weight), Case)
```




```
alpha <- 0.05
crit <- qt(1-alpha/2/nmodel1, nmodel1-p1-1)
youtlier1=which(abs(rstudent(fitmodel1weight)) >=crit )
```

```
#x outlier for model1
X <- as.matrix(cbind(rep(1,nmodel1), datamodel1[1:12]))
H <- X%*%solve(t(X)%*%X, tol=1e-20)%*%t(X)
leverage <- hatvalues(fitmodel1weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p1/nmodel1, col=2)
```



```r
xoutlier1=data.frame(which(leverage>2*p1/nmodel1) )
xoutlier1
```

```
##     which.leverage...2...p1.nmodel1.
## 4                                  4
## 31                                31
## 81                                81
## 88                                88
## 91                                91
## 104                              104
## 109                              109
## 116                              116
## 117                              117
## 121                              121
## 163                              163
## 171                              171
## 189                              189
## 197                              197
## 198                              198
## 234                              234
## 240                              240
## 241                              241
## 253                              253
## 262                              262
## 269                              269
## 293                              293
## 301                              301
## 309                              309
```

```r
#test whether outlier in the extend of the model1
IM1=influence.measures(fitmodel1weight)
dxoutlier1=union(which(IM1$infmat[,16]>0.2),which(IM1$infmat[,14]>2*sqrt(p1/nmodel1)))
#combine x and y outlier
finaloutlier1=union(dxoutlier1,youtlier1)
datamodel1Final=datamodel1[-c(finaloutlier1),]
# get model1 without x y outlier
fitmodel1x1=lm(datamodel1Final$V.9~.,data = datamodel1Final)
wtsx1 <- 1/fitted(lm(abs(residuals(fitmodel1x1)) ~ ., data = datamodel1Final))^2
Fmodel1=lm(datamodel1Final$V.9~., data = datamodel1Final,weights =wtsx1)
# R2 & adj R2 for model1
summary(Fmodel1)$r.squared
```

```
## [1] 0.9973856
```

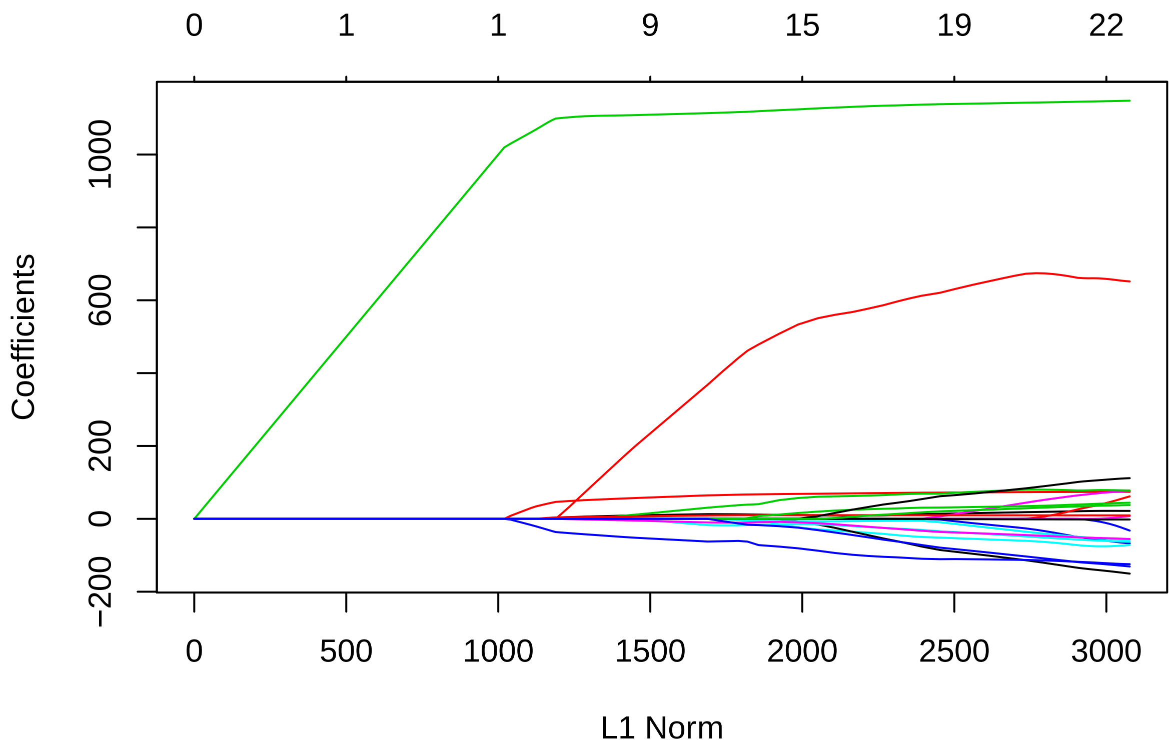
```r
summary(Fmodel1)$adj.r.squared
```

```
## [1] 0.9972828
```

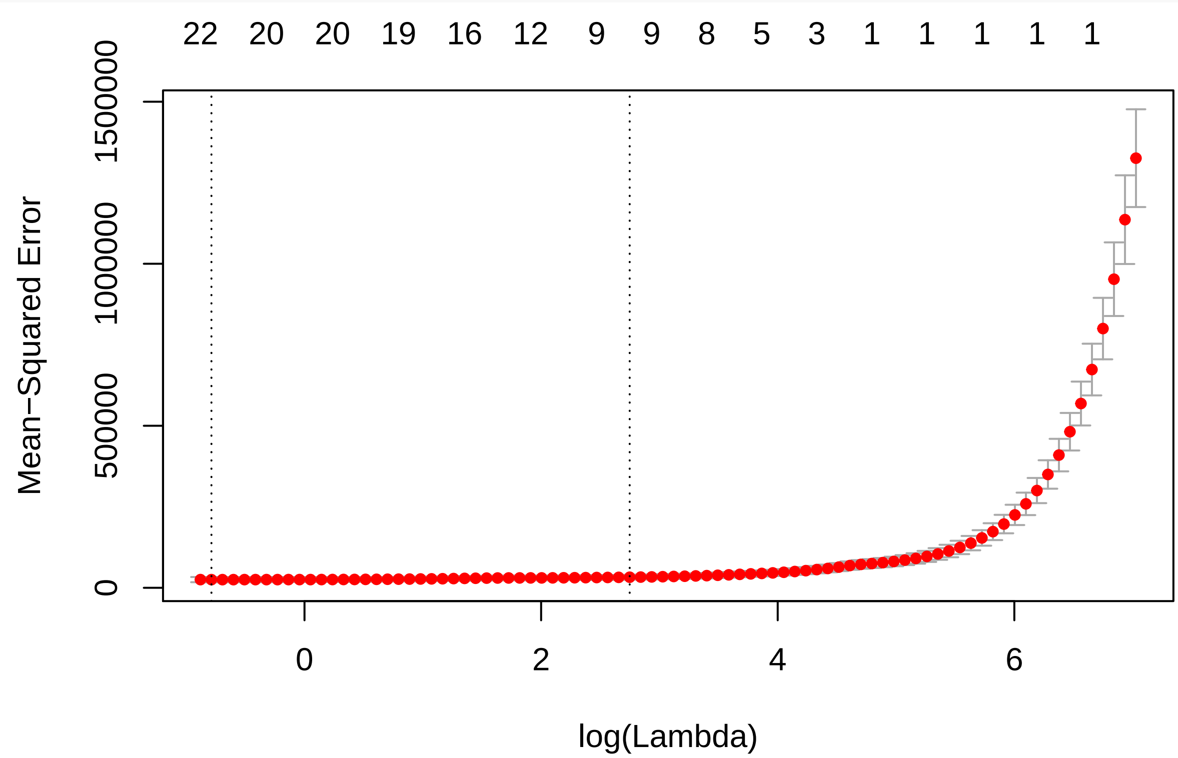
```r
# add ^2 for model2
Data.new <- cbind(train.v9$`V-4`, train.v9$`V-7`, train.v9$`V-8`, train.v9$ `V-16`, train.v9$`V-17`, tra
x2.new=as.matrix(cbind(Data.new,((Data.new)^2)[,-3]))
colnames(x2.new)=c("V-4","V-7","V-8","V-16","V-17","V-18","V-20","V-21","V-23","V-26","V-28","V-29","V-
```

```r
#lasso test x^2
library("glmnet")
fitlasso.x2add=glmnet(x2.new,y,alpha = 1)
```

```r
plot(fitlasso.x2add)
```




```r
cv.lasso.x2add=cv.glmnet(x2.new,y)
plot(cv.lasso.x2add)
```




```r
coef(cv.lasso.x2add)
```

```
## 24 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 1120.194293
## V-4           11.776093
## V-7           61.906400
## V-8         1111.793048
## V-16           .
## V-17           .
## V-18           .
## V-20           .
## V-21           .
## V-23          24.075480
## V-26           .
## V-28           .
## V-29           .
## V-4.2          .
## V-7.2          8.405836
## V-16.2         .
## V-17.2       -58.704428
## V-18.2       -11.588922
## V-20.2        -8.162791
## V-21.2         .
## V-23.2       310.170745
## V-26.2         .
## V-28.2         .
## V-29.2         .
```

```
# Model 2
trainv92 = data.frame(x2.new,y)
datamodel2=data.frame(trainv92[,c(1,2,3,9,14,16,17,18,20,24)])

fitmodel2=lm(datamodel2$y~.,data = datamodel2)
summary(fitmodel1)$r.squared
```

```
## [1] 0.9825749
```

```
summary(fitmodel1)$adj.r.squared
```

```
## [1] 0.9819235
```

```
#Brown test whether constant variance and transformation for Model 2
fitmodel2=lm(datamodel2$y~.,data = datamodel2)
resmodel2=fitmodel2$residuals
mmodel2=mean(datamodel2$y)
nmodel2=dim(datamodel2)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel2[datamodel2$y<mmodel2]
Group2 <-resmodel2[datamodel2$y>=mmodel2]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)
```

```r
#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel1-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
 t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
 t
```

```
## [1] -5.581285
```

```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)   # find the catical value
```

```
## [1] 1.967405
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel1-p1-1))
```

```
## [1] 5.095215e-08
```

```r
# Weighted tranformation for model 2
wts <- 1/fitted(lm(abs(residuals(fitmodel2)) ~ ., data = datamodel2))^2

fitmodel2weight <- lm(datamodel2$y~ .,data = datamodel2, weights=wts)
datamodel2weight=cbind(datamodel2[1:9],datamodel2$y*wts)
summary(fitmodel2weight)$r.squared
```

```
## [1] 0.9897624
```

```r
summary(fitmodel2weight)$adj.r.squared
```

```
## [1] 0.989478
```

```r
#Brown test whether constant variance and transformation for Model 2 after tranformation
resmodel2b=fitmodel2weight$residuals
mmodel2=mean(datamodel2weight$`datamodel2$y * wts`)
nmodel2=dim(datamodel2weight)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel2b[datamodel2weight$`datamodel2$y * wts`<mmodel2]
Group2 <-resmodel2b[datamodel2weight$`datamodel2$y * wts`>=mmodel2]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel2-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
t
```

```
## [1] 0.725008
```

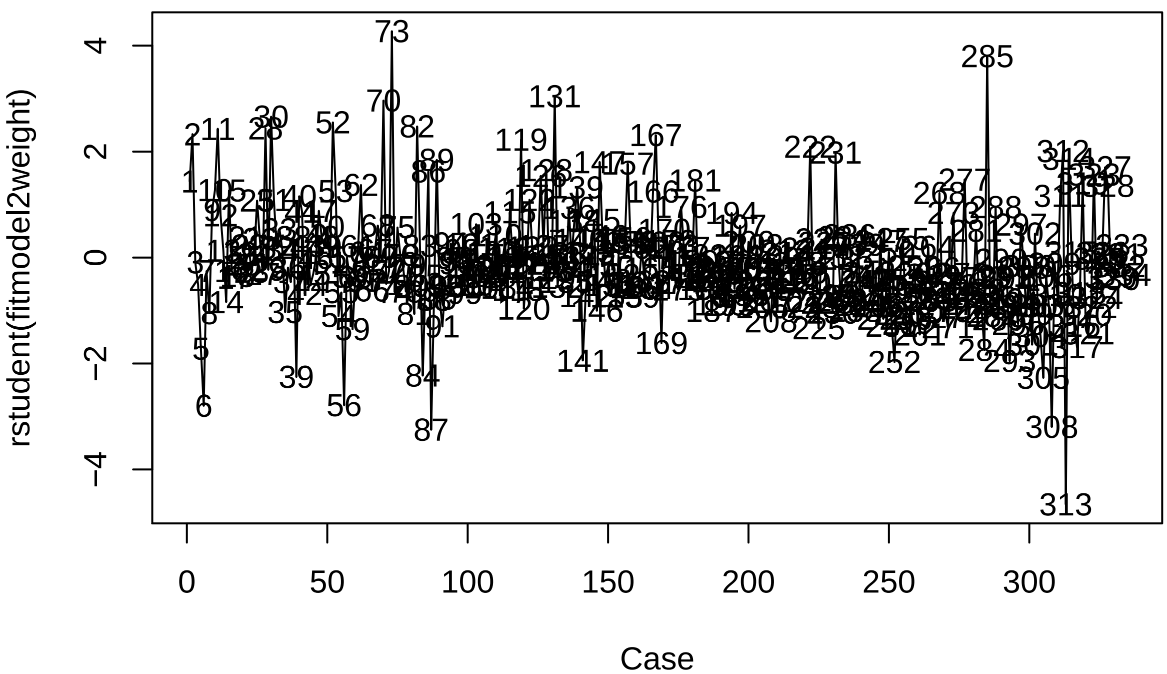
```
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel2-17)   # find the catical value
```

```
## [1] 1.967476
```

```
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel2-17))
```

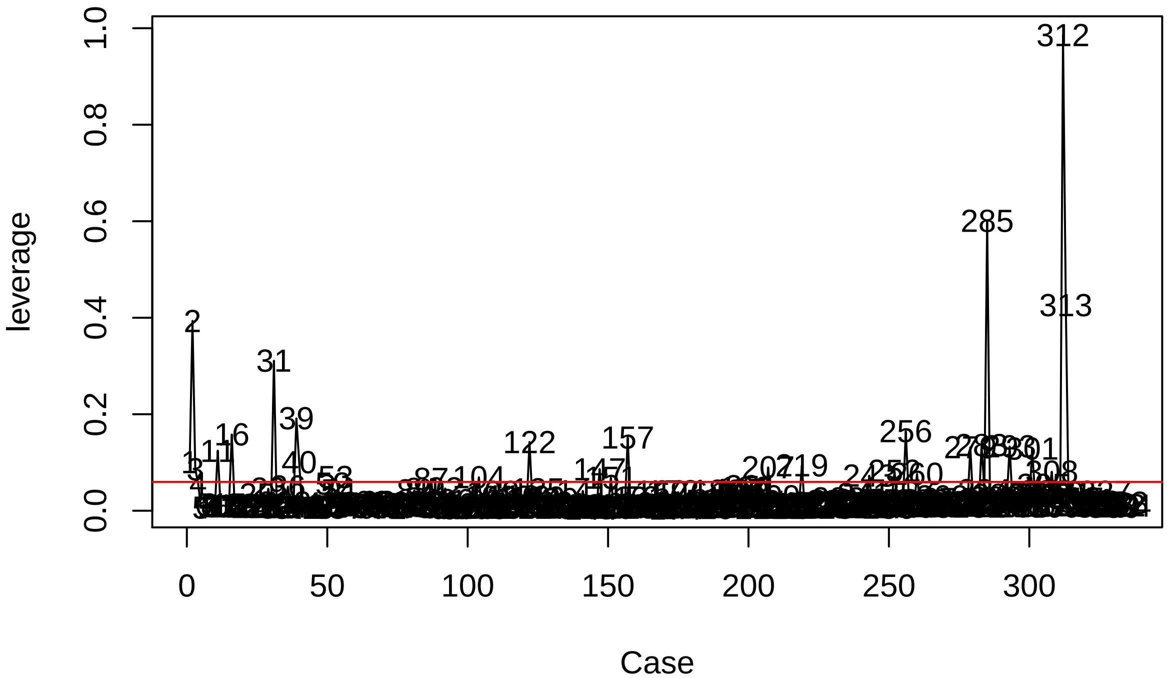
```
## [1] 0.4689819
```

```
#y outlier
Case <- c(1:nmodel2)
plot(Case, rstudent(fitmodel2weight), type="l")
text(Case, rstudent(fitmodel2weight), Case)
```




```
alpha <- 0.01
p=10
crit <- qt(1-alpha/2/nmodel2, nmodel2-p-1)
youtlier=which(abs(rstudent(fitmodel2weight)) >=crit )
```

```
#x outlier
X <- as.matrix(cbind(rep(1,nmodel2), datamodel2weight[1:9]))
H <- X%*%solve(t(X)%*%X,tol=1e-30)%*%t(X)
leverage <- hatvalues(fitmodel2weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p/nmodel2, col=2)
```




```
xoutlier=data.frame(which(leverage>2*p/nmodel2) )
xoutlier
```

```
##     which.leverage...2...p.nmodel2.
## 1                                 1
## 2                                 2
## 3                                 3
## 11                               11
## 16                               16
## 31                               31
## 39                               39
## 40                               40
## 53                               53
## 87                               87
## 104                             104
## 122                             122
## 147                             147
## 151                             151
## 157                             157
## 207                             207
## 219                             219
## 243                             243
## 252                             252
## 256                             256
## 260                             260
## 279                             279
## 283                             283
## 285                             285
## 293                             293
## 301                             301
## 308                             308
## 312                             312
## 313                             313
```

```r
#test whether outlier in the extend of the model
IM2=influence.measures(fitmodel2weight)
dxoutlier=union(which(IM2$infmat[,13]>0.2),which(IM2$infmat[,11]>2*sqrt(p/nmodel2)))
#combine x and y outlier
finaloutlier=union(dxoutlier,youtlier)
datamodel2Final=datamodel2[-c(finaloutlier),]
# get model2 without x y outlier
fitmodel2x2=lm(datamodel2Final$y~.,data = datamodel2Final)
wtsx2 <- 1/fitted(lm(abs(residuals(fitmodel2x2)) ~ ., data = datamodel2Final))^2
Fmodel2=lm(datamodel2Final$y~., data = datamodel2Final,weights =wtsx2)
# R2 & adj R2 for model1
summary(Fmodel2)$r.squared
```

```
## [1] 0.9955124
```

```r
summary(Fmodel2)$adj.r.squared
```

```
## [1] 0.9953821
```

```r
#VIF
# model 1
data1Finalvif=datamodel1Final[,-13]
vif1=rep(0:12)
vif1[1]=1/(1-summary(lm(data1Finalvif$V.4~ .,data = data1Finalvif))$r.squared)
vif1[2]=1/(1-summary(lm(data1Finalvif$V.7~.,data = data1Finalvif))$r.squared)
vif1[3]=1/(1-summary(lm(data1Finalvif$V.8~.,data = data1Finalvif))$r.squared)
vif1[4]=1/(1-summary(lm(data1Finalvif$V.16~.,data = data1Finalvif))$r.squared)
vif1[5]=1/(1-summary(lm(data1Finalvif$V.17~.,data = data1Finalvif))$r.squared)
vif1[6]=1/(1-summary(lm(data1Finalvif$V.18~.,data = data1Finalvif))$r.squared)
vif1[7]=1/(1-summary(lm(data1Finalvif$V.20~.,data = data1Finalvif))$r.squared)
vif1[8]=1/(1-summary(lm(data1Finalvif$V.21~.,data = data1Finalvif))$r.squared)
vif1[9]=1/(1-summary(lm(data1Finalvif$V.23~.,data = data1Finalvif))$r.squared)
vif1[10]=1/(1-summary(lm(data1Finalvif$V.26~.,data = data1Finalvif))$r.squared)
vif1[11]=1/(1-summary(lm(data1Finalvif$V.28~.,data = data1Finalvif))$r.squared)
vif1[12]=1/(1-summary(lm(data1Finalvif$V.29~.,data = data1Finalvif))$r.squared)
vif1
```

```
##  [1]  1.343717  1.085832  2.021489 12.994107 49.417724  6.649542  2.153869
##  [8] 56.258462  7.233167 90.247532  5.222142 24.729647 12.000000
```

```r
#model2
data2Finalvif=datamodel2Final[,-10]
vif2=rep(0:9)
vif2[1]=1/(1-summary(lm(data2Finalvif$V.4~ .,data = data2Finalvif))$r.squared)
vif2[2]=1/(1-summary(lm(data2Finalvif$V.7~ .,data = data2Finalvif))$r.squared)
vif2[3]=1/(1-summary(lm(data2Finalvif$V.8~ .,data = data2Finalvif))$r.squared)
vif2[4]=1/(1-summary(lm(data2Finalvif$V.23~ .,data = data2Finalvif))$r.squared)
vif2[5]=1/(1-summary(lm(data2Finalvif$V.7.2~ .,data = data2Finalvif))$r.squared)
vif2[6]=1/(1-summary(lm(data2Finalvif$V.17.2~ .,data = data2Finalvif))$r.squared)
vif2[7]=1/(1-summary(lm(data2Finalvif$V.18.2~ .,data = data2Finalvif))$r.squared)
vif2[8]=1/(1-summary(lm(data2Finalvif$V.20.2~ .,data = data2Finalvif))$r.squared)
vif2[9]=1/(1-summary(lm(data2Finalvif$V.23.2~ .,data = data2Finalvif))$r.squared)

vif2
```

```
##  [1] 1.279930 1.431435 1.786355 1.566845 1.440946 2.614020 1.122960
```

```
##  [8] 2.285561 1.399707 9.000000
```

```
#test the model
# Import the data and pretreatment
load("~/Desktop/Study in NU/Winter/Regression analysis/Final/train & test.RData")
#strandized for test data
yt9=as.matrix(test$`V-9`)
colnames(yt9)=c("V-9")
yt10=as.matrix(test$`V-10`)
colnames(yt10)=c("V-10")
test.v9<- cbind(test[1:27],yt9)
test.v10<- cbind(test[1:27],yt10)
for(i in 2:27)
 {
  test.v9[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
for(i in 2:27)
 {
  test.v10[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
```

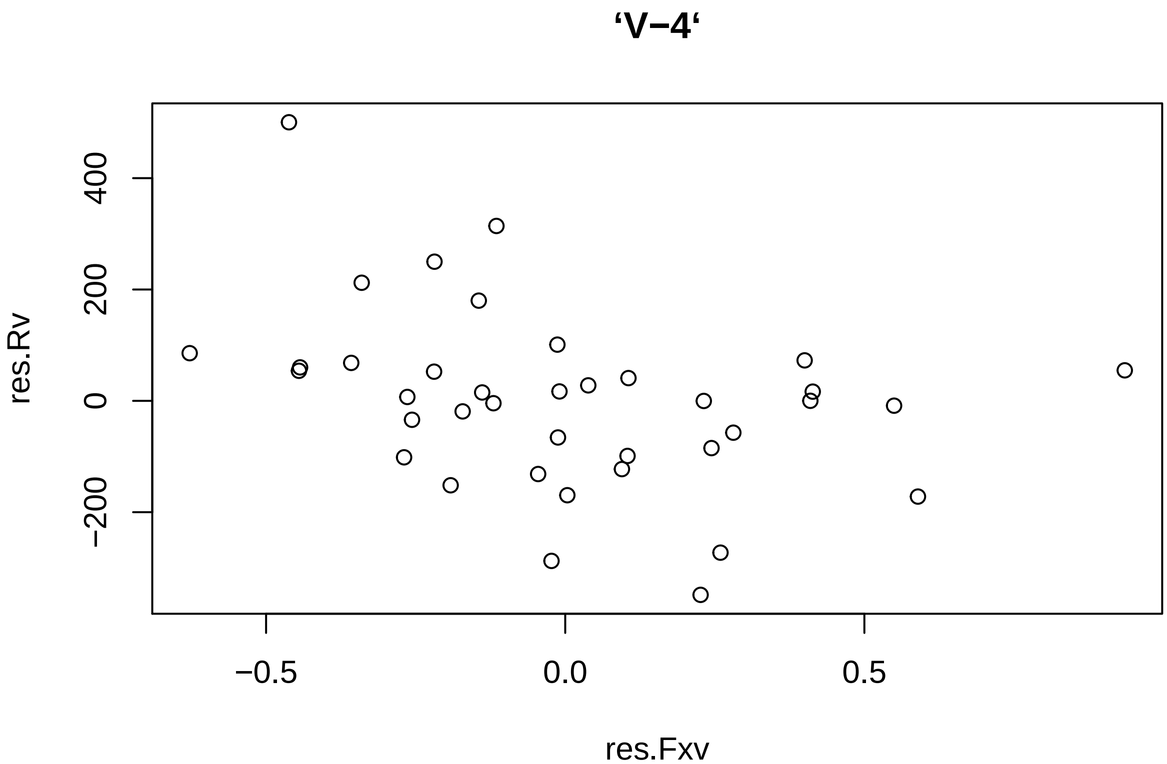
```
# added variable factor to determine ^
datamodel1=data.frame(test.v9[,c(4,7,8,14,15,16,18,19,21,24,26,27,28)])
# Model 1
fitmodel1=lm(datamodel1$V.9~.,data = datamodel1)
summary(fitmodel1)$r.squared
```

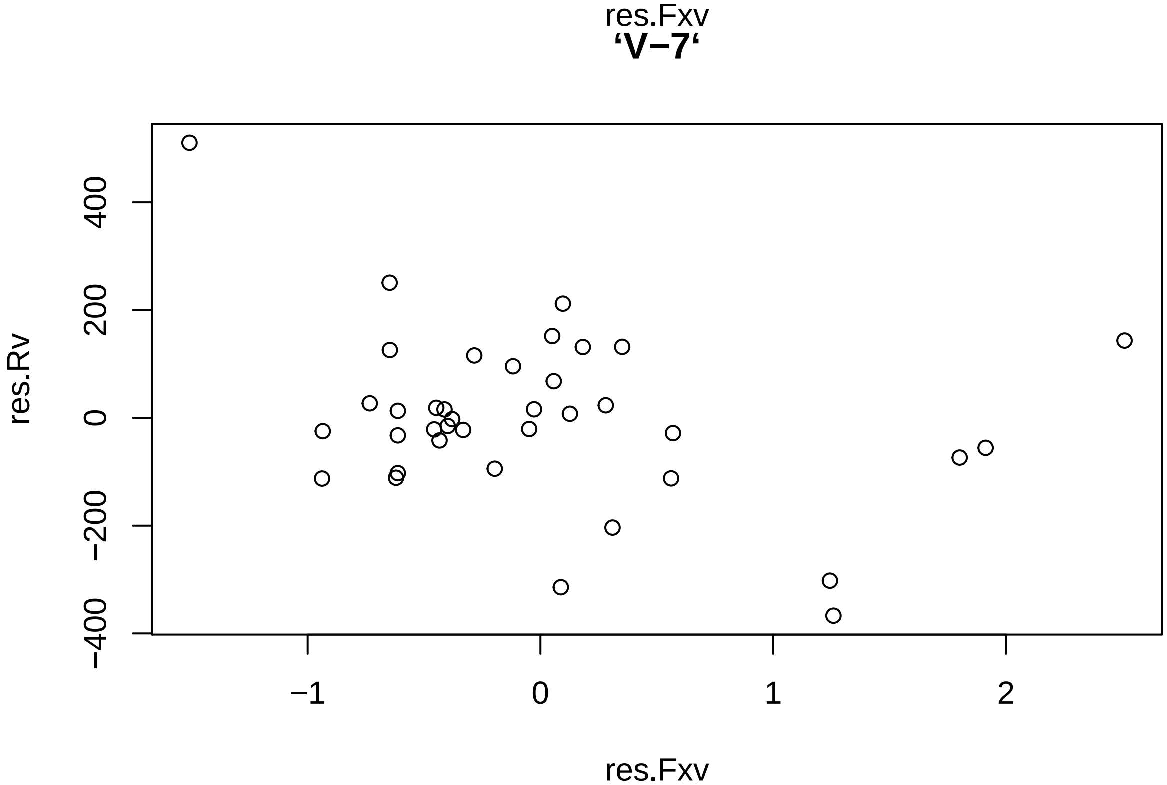
```
## [1] 0.9909658
```

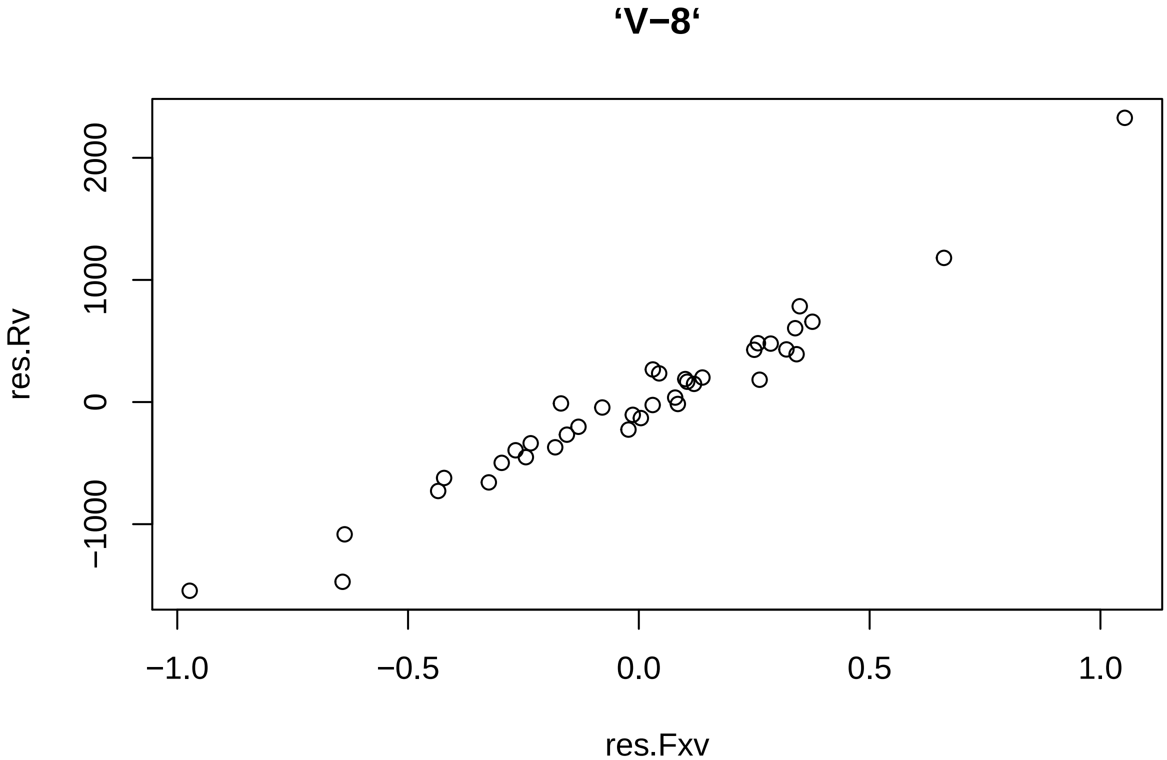
```
summary(fitmodel1)$adj.r.squared
```

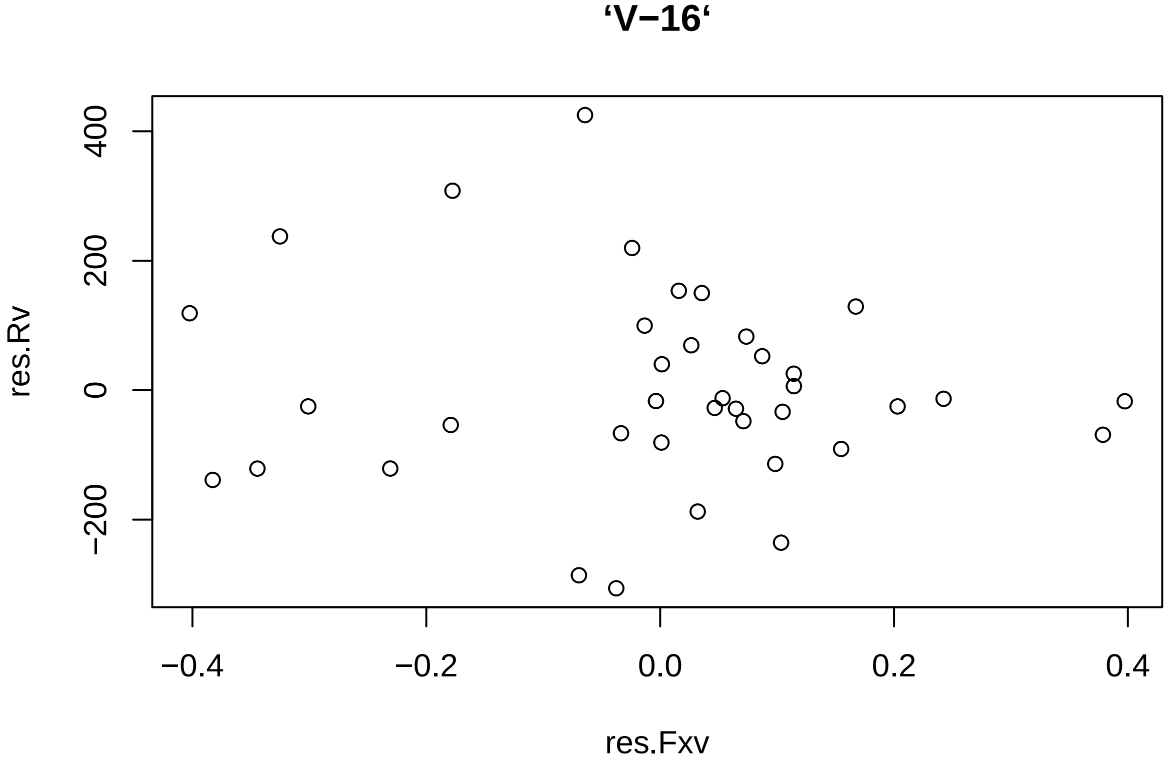
```
## [1] 0.9866294
```

```
colData <- list("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
names(colData) <- c("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
removeXList <- colData

for (rmX in removeXList){
  tmpV <- colData
  tmpV[[rmX]] = NULL
  test.Rv=lm(as.formula(paste("`V-9` ~", paste(tmpV, collapse = "+"))), data = test.v9)
  res.Rv= test.Rv$residuals

  test.Fxv=lm(as.formula(paste(paste(rmX," ~"), paste(tmpV, collapse = "+"))), data = test.v9)
  res.Fxv= test.Fxv$residuals

  plot(res.Fxv,res.Rv,main = rmX)
}
```

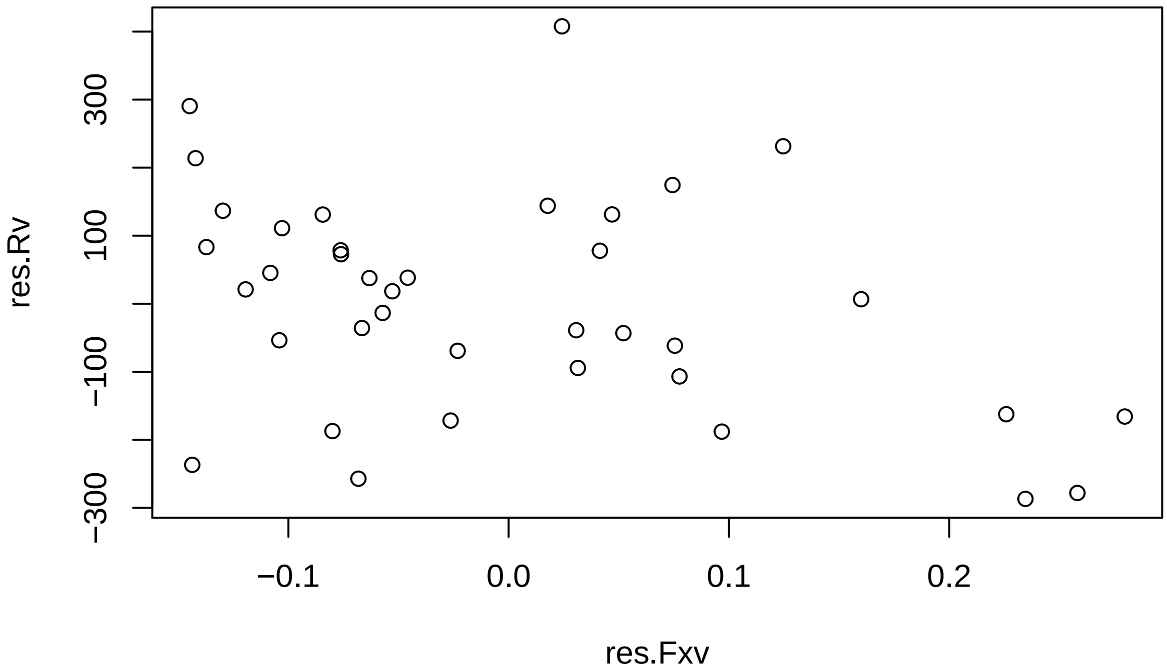

‘V−4‘
res.Rv
res.Fxv
400
200
0
−200
−0.5
0.0
0.5


‘V−7‘
res.Rv
res.Fxv
400
200
0
−200
−400
−1
0
1
2


‘V−8‘
res.Rv
res.Fxv
−1000
0
1000
2000
−1.0
−0.5
0.0
0.5
1.0

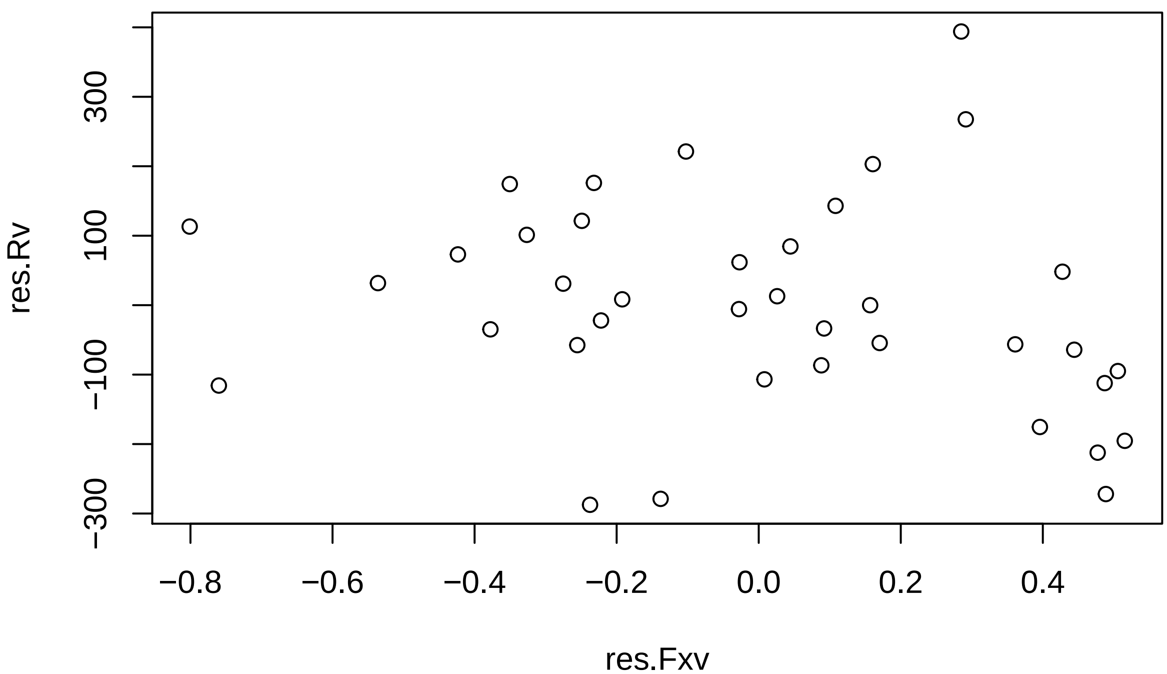

‘V−16‘
res.Rv
res.Fxv
−200
0
200
400
−0.4
−0.2
0.0
0.2
0.4

‘V−17‘

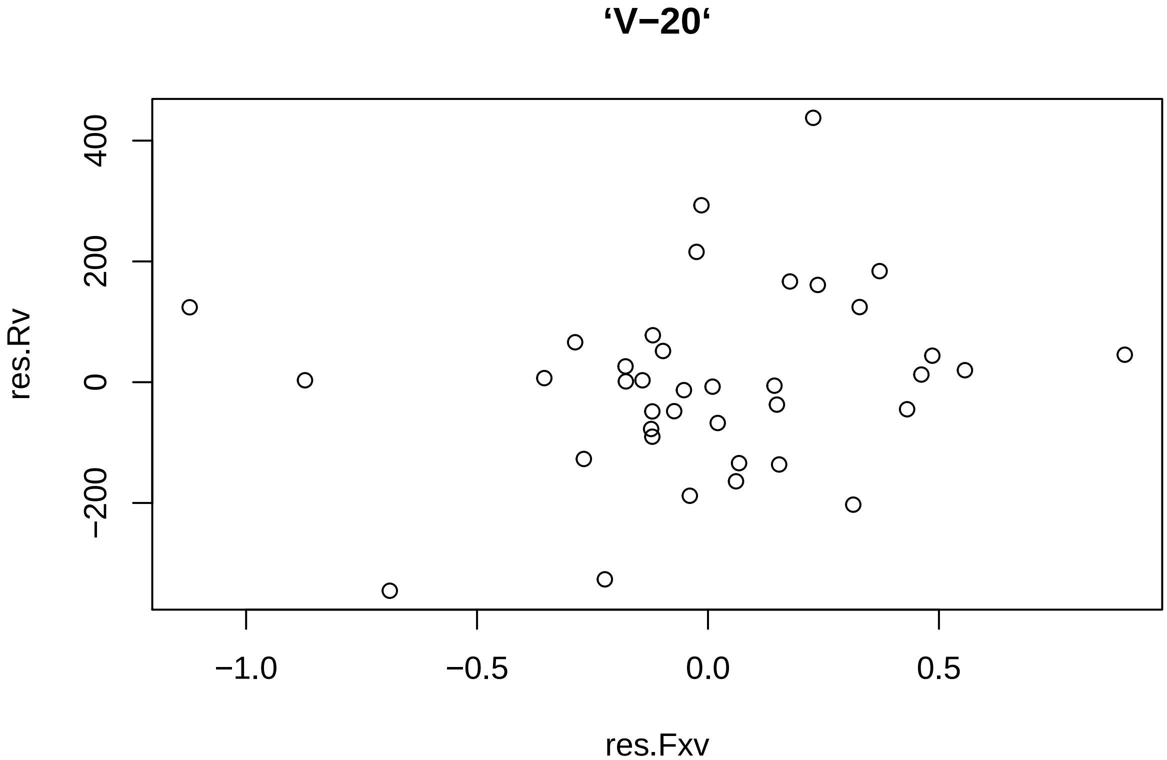


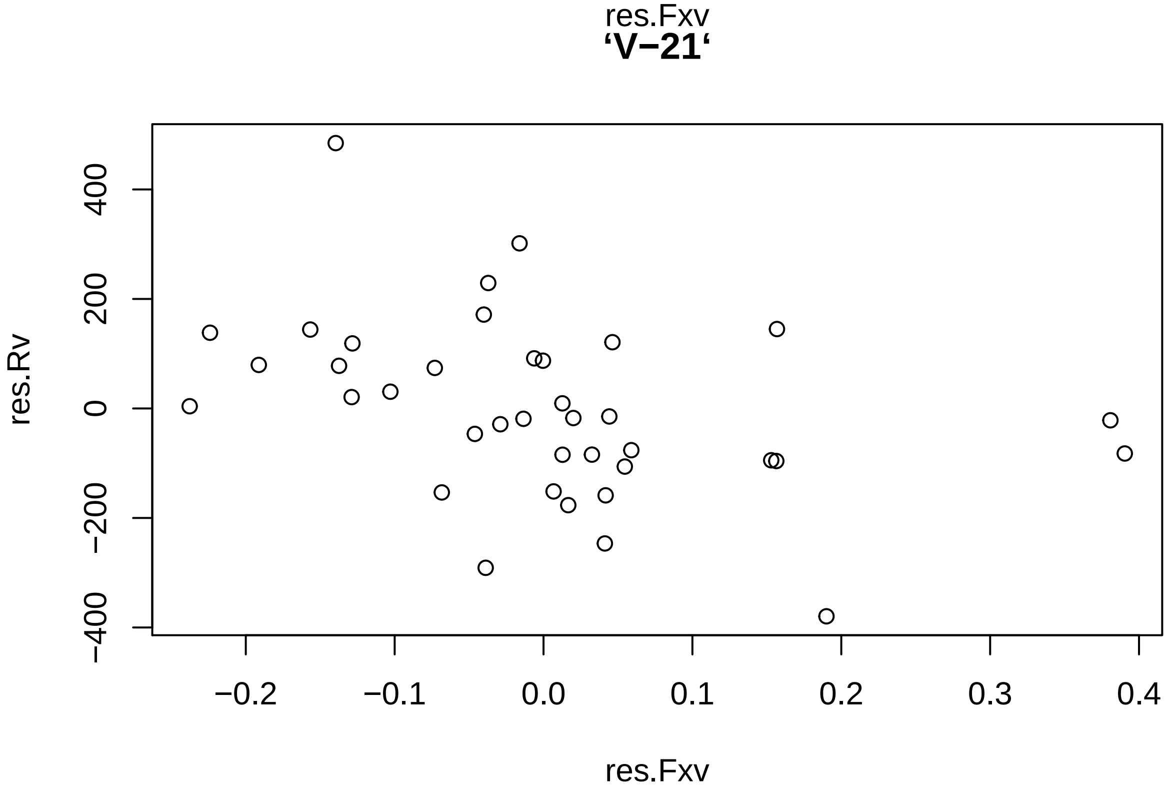
res.Rv
res.Fxv
−300
−100
100
300
−0.1
0.0
0.1
0.2

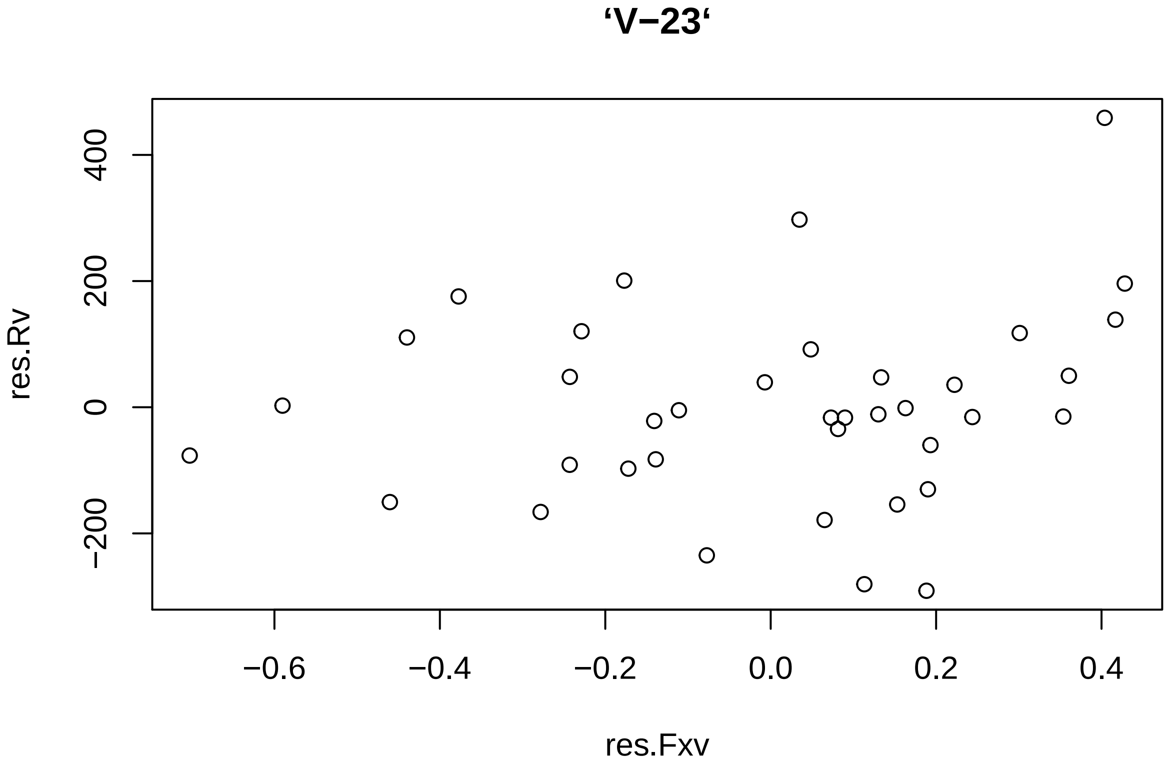
‘V−18‘

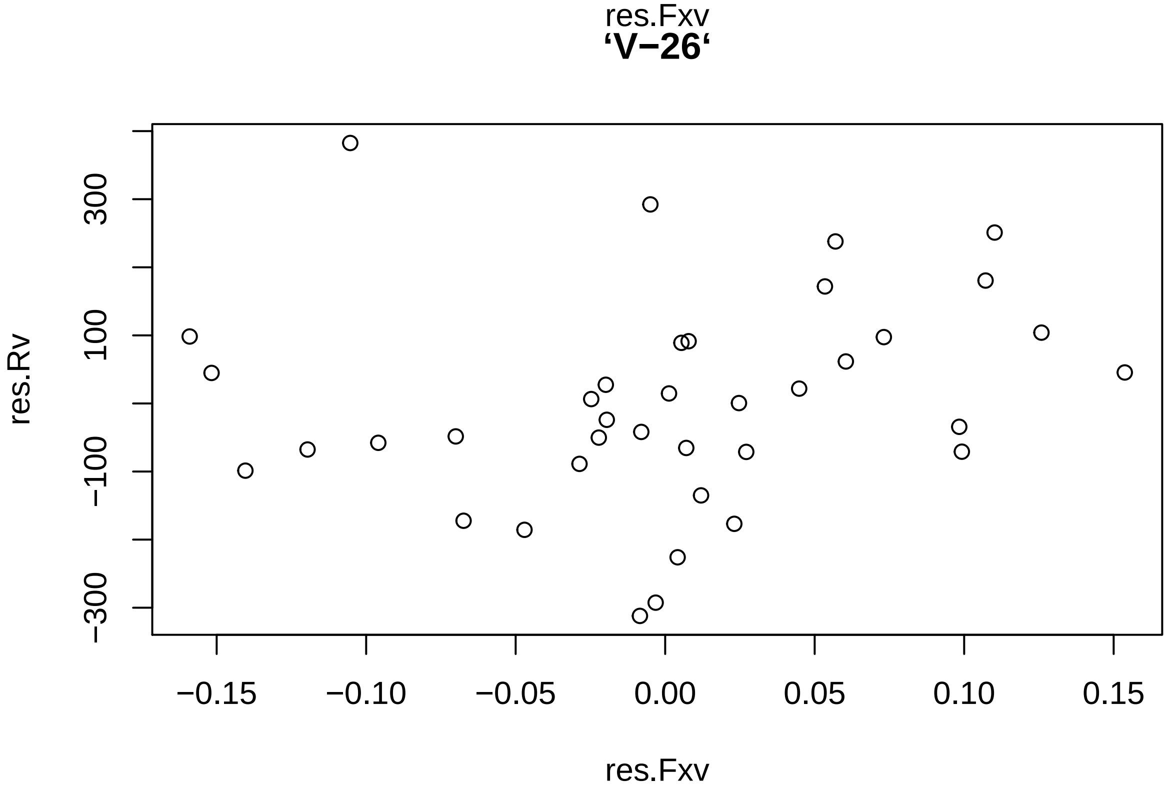


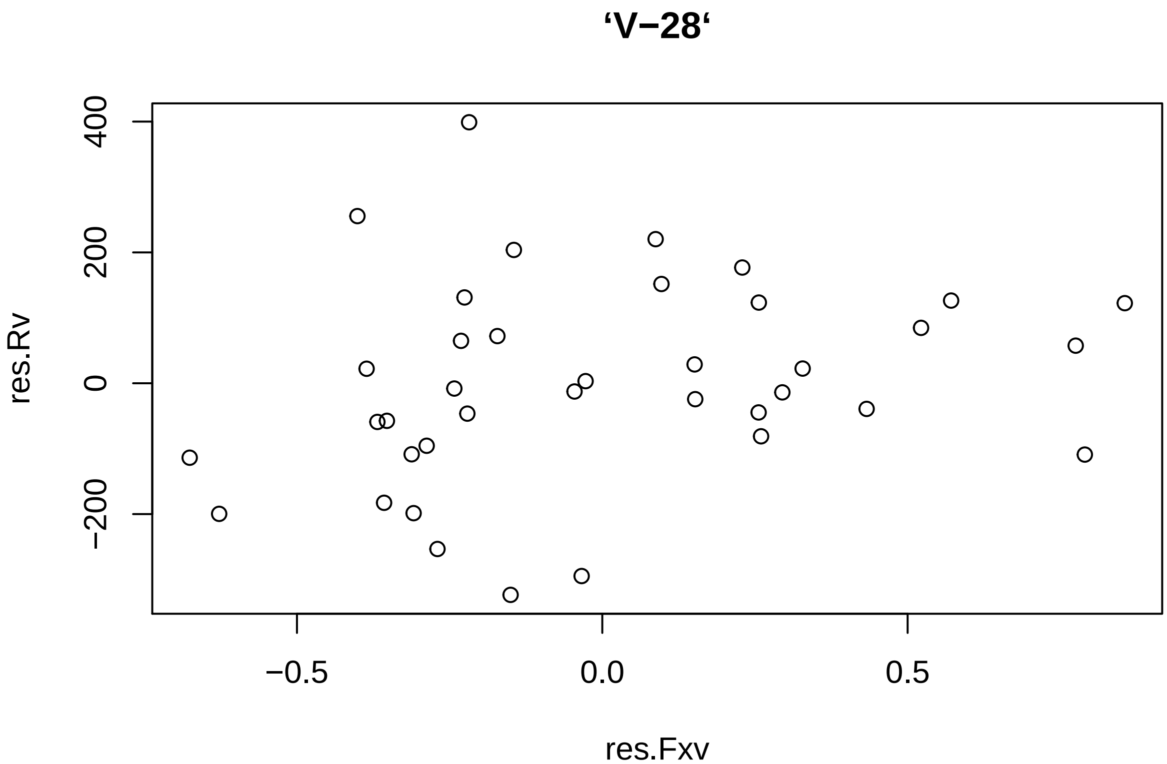
res.Rv
res.Fxv
−300
−100
100
300
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4

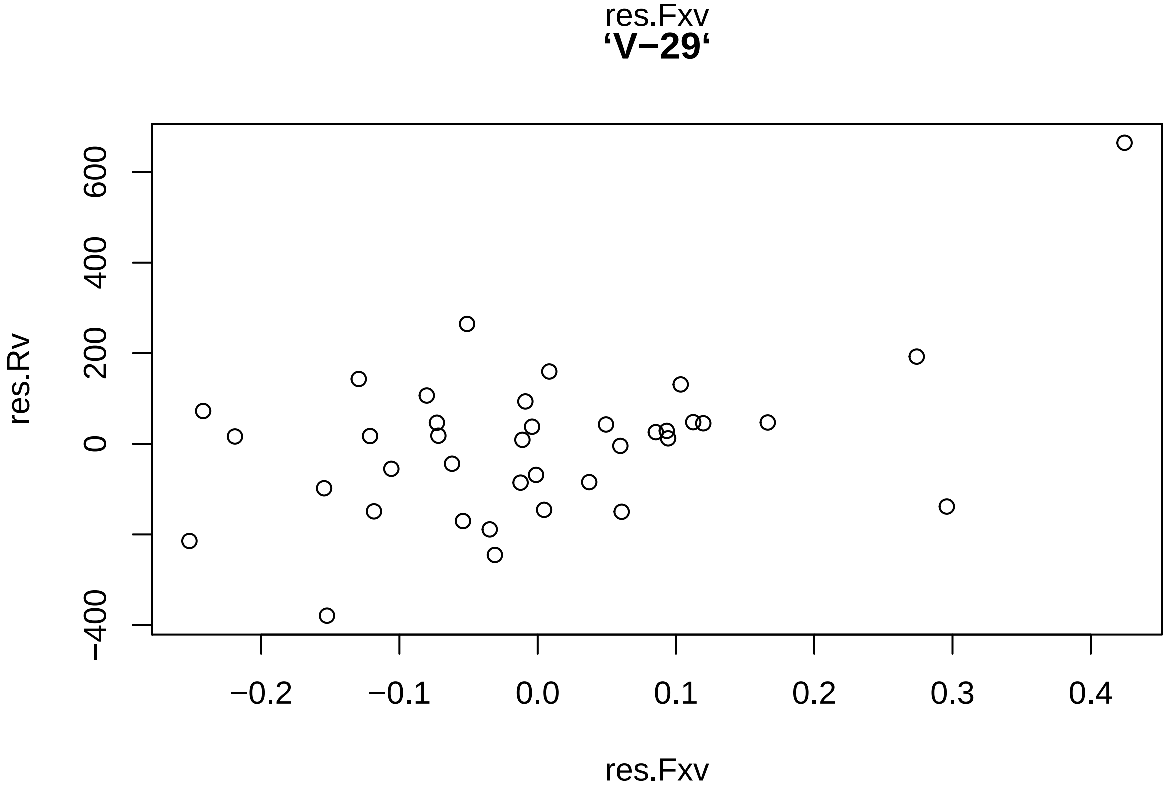

‘V−20‘
res.Rv
400
200
0
−200
−1.0
−0.5
0.0
0.5
res.Fxv


‘V−21‘
res.Rv
400
200
0
−200
−400
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
res.Fxv


‘V−23‘
res.Rv
−200
0
200
400
−0.6
−0.4
−0.2
0.0
0.2
0.4
res.Fxv


‘V−26‘
res.Rv
−300
−100
100
300
−0.15
−0.10
−0.05
0.00
0.05
0.10
0.15
res.Fxv






```
#Brown test whether constant variance and transformation for Model 1
resmodel1=fitmodel1$residuals
mmodel1=mean(datamodel1$V.9)
nmodel1=dim(datamodel1)[1]
p1=13
#1. Break the residuals into two groups.
```

```
Group1 <- resmodel1[datamodel1$V.9<mmodel1]
Group2 <-resmodel1[datamodel1$V.9>=mmodel1]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel1-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
 t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
 t
```

```
## [1] -0.7239271
```

```
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)   # find the catical value
```

```
## [1] 2.063899
```

```
# Weighted tranformation for model 1
wts <- 1/fitted(lm(abs(residuals(fitmodel1)) ~ ., data = datamodel1))^2

fitmodel1weight <- lm(datamodel1$V.9~ .,data = datamodel1, weights=wts)
datamodel1weight=cbind(datamodel1[1:12],datamodel1$V.9*wts)
summary(fitmodel1weight)$r.squared
```

```
## [1] 0.9995159
```

```
summary(fitmodel1weight)$adj.r.squared
```

```
## [1] 0.9992836
```

```
#Brown test whether constant variance and transformation for Model 1 after tranformation
resmodel1b=fitmodel1weight $residuals
mmodel1=mean(datamodel1weight$`datamodel1$V.9 * wts`)
nmodel1=dim(datamodel1weight)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel1b[datamodel1weight$`datamodel1$V.9 * wts`<mmodel1]
Group2 <-resmodel1b[datamodel1weight$`datamodel1$V.9 * wts`>=mmodel1]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
```

```
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel1-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
t
```

```
## [1] 1.301284
```

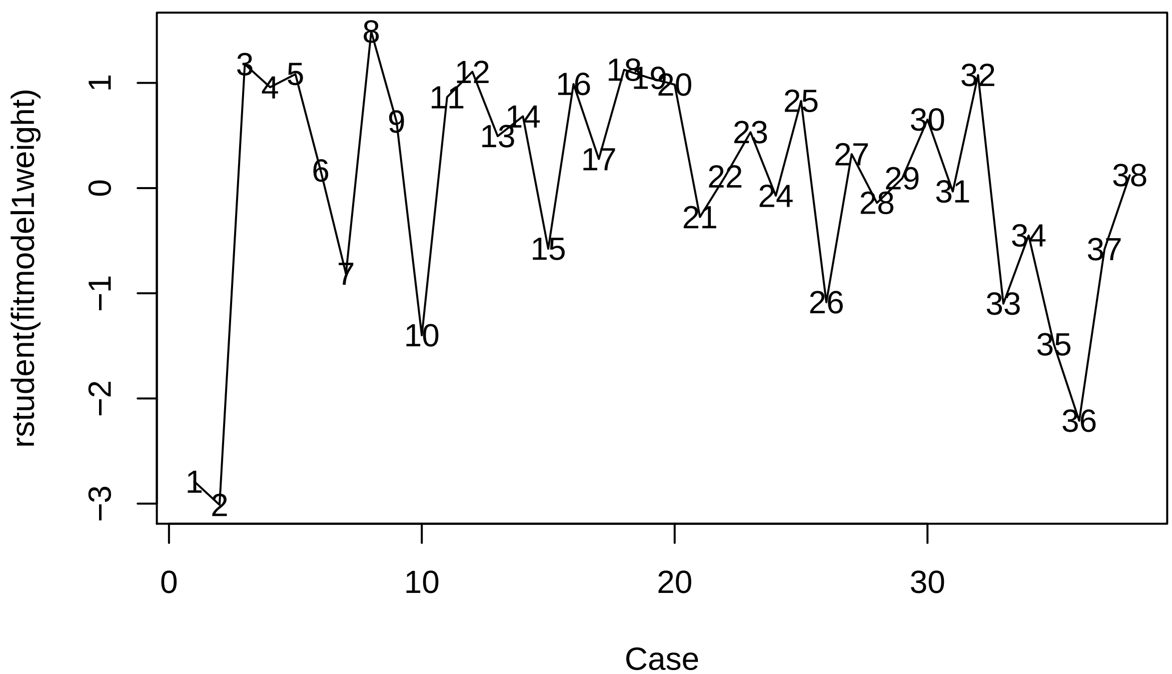
```
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)    # find the catical value
```

```
## [1] 2.063899
```

```
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel1-p1-1))
```

```
## [1] 0.2055156
```

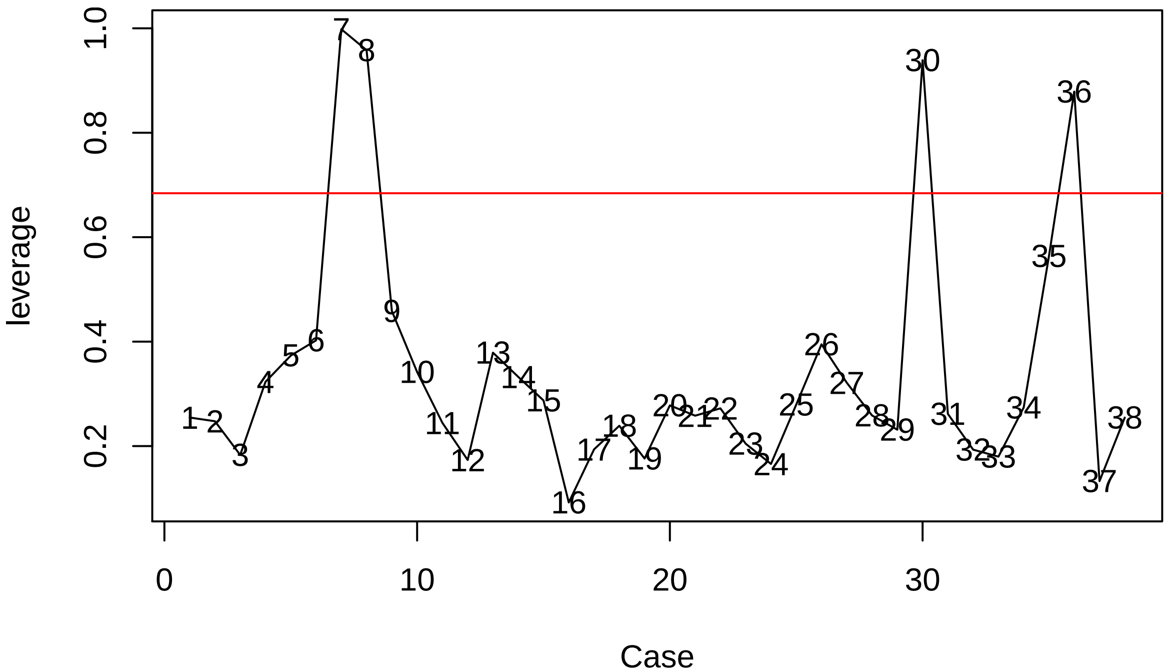
```
#y outlier for model1
Case <- c(1:nmodel1)
plot(Case, rstudent(fitmodel1weight), type="l")
text(Case, rstudent(fitmodel1weight), Case)
```




```
alpha <- 0.05
crit <- qt(1-alpha/2/nmodel1, nmodel1-p1-1)
youtlier1=which(abs(rstudent(fitmodel1weight)) >=crit )
```

```
#x outlier for model1
X <- as.matrix(cbind(rep(1,nmodel1), datamodel1[1:12]))
H <- X%*%solve(t(X)%*%X, tol=1e-20)%*%t(X)
leverage <- hatvalues(fitmodel1weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
```

```
abline(h=2*p1/nmodel1, col=2)
```




```
xoutlier1=data.frame(which(leverage>2*p1/nmodel1) )
xoutlier1
```

```
##    which.leverage...2...p1.nmodel1.
## 7                                7
## 8                                8
## 30                              30
## 36                              36
```

```
#test whether outlier in the extend of the model1
IM1=influence.measures(fitmodel1weight)
dxoutlier1=union(which(IM1$infmat[,16]>0.2),which(IM1$infmat[,14]>2*sqrt(p1/nmodel1)))
#combine x and y outlier
finaloutlier1=union(dxoutlier1,youtlier1)
datamodel1Final=datamodel1[-c(finaloutlier1),]
# get model1 without x y outlier
fitmodel1x1=lm(datamodel1Final$V.9~.,data = datamodel1Final)
wtsx1 <- 1/fitted(lm(abs(residuals(fitmodel1x1)) ~ ., data = datamodel1Final))^2
Fmodel1=lm(datamodel1Final$V.9~., data = datamodel1Final,weights =wtsx1)
# R2 & adj R2 for model1 test
summary(Fmodel1)$r.squared
```

```
## [1] 0.9986705
```

```
summary(Fmodel1)$adj.r.squared
```

```
## [1] 0.9978728
```

```
# add ^2 for model1
Data.new <- cbind(test.v9$`V-4`, test.v9$`V-7`, test.v9$`V-8`, test.v9$ `V-16`, test.v9$`V-17`, test.v9
x2.new=as.matrix(cbind(Data.new,((Data.new)^2)[,-3]))
colnames(x2.new)=c("V-4","V-7","V-8","V-16","V-17","V-18","V-20","V-21","V-23","V-26","V-28","V-29","V-
```

```
# Model 2
y=test.v9$`V-9`
testv92 = data.frame(x2.new,y)
datamodel2=data.frame(testv92[,c(1,2,3,9,14,16,17,18,20,24)])

fitmodel2=lm(datamodel2$y~.,data = datamodel2)
summary(fitmodel1)$r.squared
```

```
## [1] 0.9909658
```

```
summary(fitmodel1)$adj.r.squared
```

```
## [1] 0.9866294
```

```
# Weighted tranformation for model 2
wts <- 1/fitted(lm(abs(residuals(fitmodel2)) ~ ., data = datamodel2))^2

fitmodel2weight <- lm(datamodel2$y~ .,data = datamodel2, weights=wts)
datamodel2weight=cbind(datamodel2[1:9],datamodel2$y*wts)
summary(fitmodel2weight)$r.squared
```

```
## [1] 0.9919199
```

```
summary(fitmodel2weight)$adj.r.squared
```

```
## [1] 0.9893227
```

```
#Brown test whether constant variance and transformation for Model 2 after tranformation
resmodel2b=fitmodel2weight$residuals
mmodel2=mean(datamodel2weight$`datamodel2$y * wts`)
nmodel2=dim(datamodel2weight)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel2b[datamodel2weight$`datamodel2$y * wts`<mmodel2]
Group2 <-resmodel2b[datamodel2weight$`datamodel2$y * wts`>=mmodel2]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel2-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
t
```

```
## [1] 2.045055
```

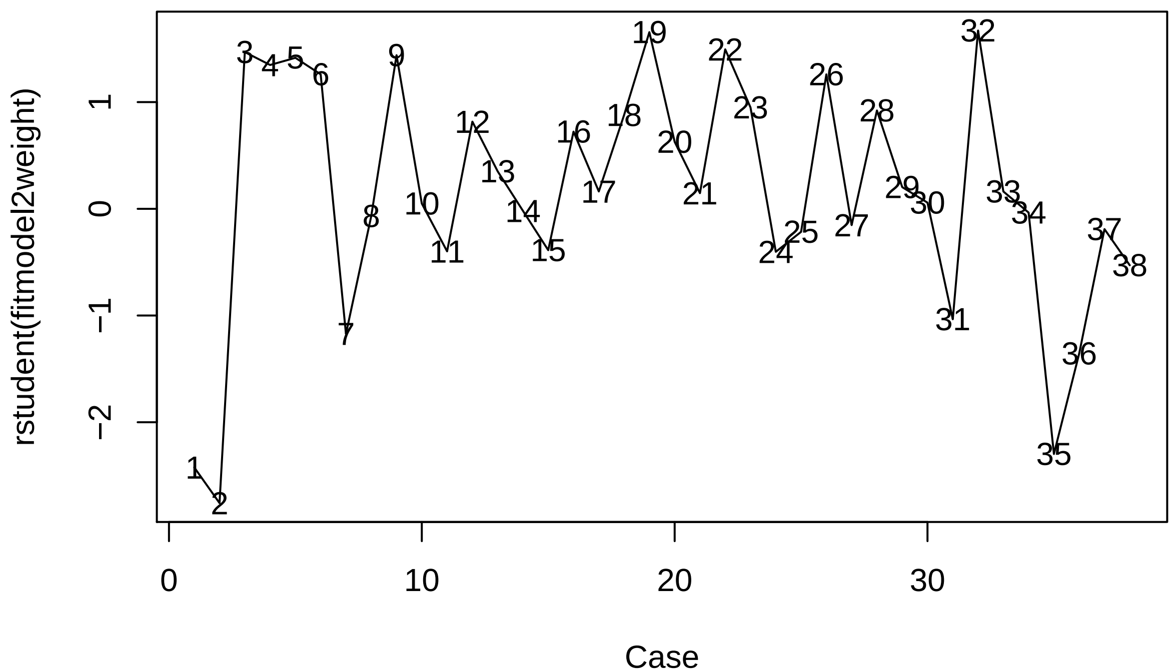
```
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel2-17)   # find the catical value
```

```
## [1] 2.079614
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel2-17))
```

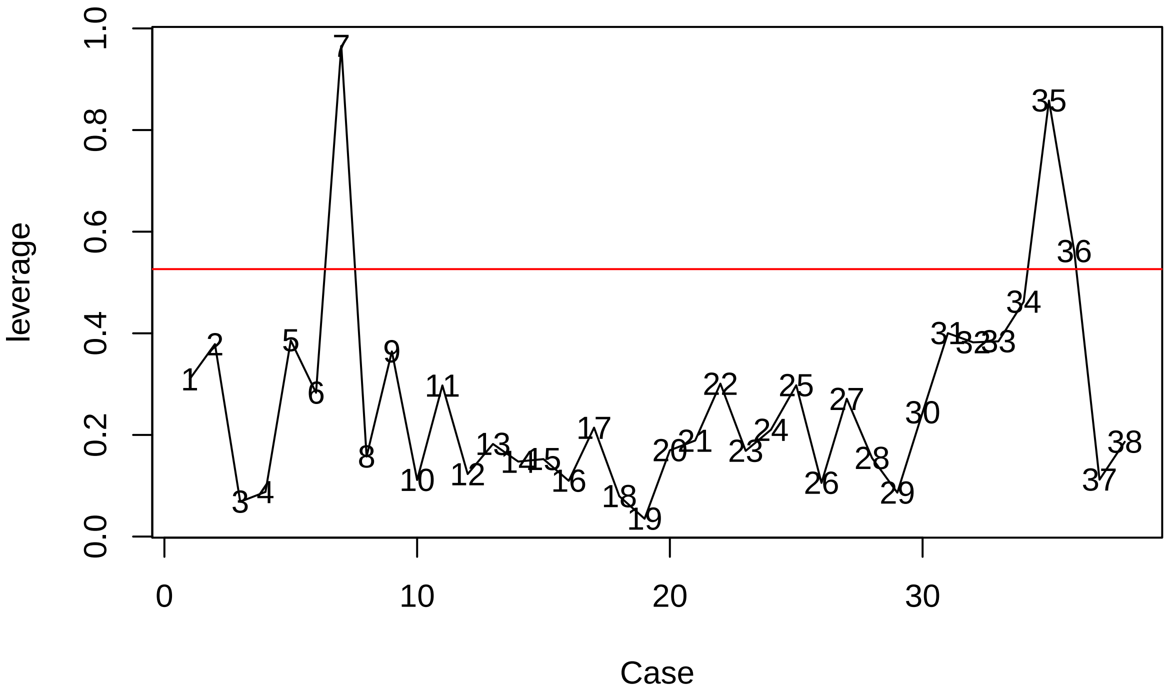
```
## [1] 0.05358353
```

```r
#y outlier
Case <- c(1:nmodel2)
plot(Case, rstudent(fitmodel2weight), type="l")
text(Case, rstudent(fitmodel2weight), Case)
```




```r
alpha <- 0.01
p=10
crit <- qt(1-alpha/2/nmodel2, nmodel2-p-1)
youtlier=which(abs(rstudent(fitmodel2weight)) >=crit )
```

```r
#x outlier
X <- as.matrix(cbind(rep(1,nmodel2), datamodel2weight[1:9]))
H <- X%*%solve(t(X)%*%X,tol=1e-30)%*%t(X)
leverage <- hatvalues(fitmodel2weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p/nmodel2, col=2)
```




```
xoutlier=data.frame(which(leverage>2*p/nmodel2) )
xoutlier
```

```
##    which.leverage...2...p.nmodel2.
## 7                                7
## 35                              35
## 36                              36
```

```
#test whether outlier in the extend of the model
IM2=influence.measures(fitmodel2weight)
dxoutlier=union(which(IM2$infmat[,13]>0.2),which(IM2$infmat[,11]>2*sqrt(p/nmodel2)))
#combine x and y outlier
finaloutlier=union(dxoutlier,youtlier)
datamodel2Final=datamodel2[-c(finaloutlier),]
# get model2 without x y outlier
fitmodel2x2=lm(datamodel2Final$y~.,data = datamodel2Final)
wtsx2 <- 1/fitted(lm(abs(residuals(fitmodel2x2)) ~ ., data = datamodel2Final))^2
Fmodel2=lm(datamodel2Final$y~., data = datamodel2Final,weights =wtsx2)
# R2 & adj R2 for model2
summary(Fmodel2)$r.squared
```

```
## [1] 0.9962238
```

```
summary(Fmodel2)$adj.r.squared
```

```
## [1] 0.9945245
```

STAT 350 Final Project of predicting V-10

Xiaotian Ding, Jie Gu, De Lu, Huaiyi Liu

2019/3/10

```r
# Import the data and pretreatment
load("~/Desktop/Study in NU/Winter/Regression analysis/Final/train & test.RData")
Zipcode= as.factor(train$`V-1`)
Zipcodetest=as.factor(test$`V-1`)
#strandized for train
y9=as.matrix(train$`V-9`)
colnames(y9)=c("V-9")
y10=as.matrix(train$`V-10`)
colnames(y10)=c("V-10")
train.v9<- cbind(Zipcode,train[2:27],y9)
train.v10<- cbind(Zipcode,train[2:27],y10)
for(i in 2:27)
 {
  train.v9[,i] <-(train[,i]-mean(train[,i])) /sd(train[,i])
}
for(i in 2:27)
 {
  train.v10[,i] <-(train[,i]-mean(train[,i])) /sd(train[,i])
}

#strandized for test data
yt9=as.matrix(test$`V-9`)
colnames(yt9)=c("V-9")
yt10=as.matrix(test$`V-10`)
colnames(yt10)=c("V-10")
test.v9<- cbind(test[1:27],yt9)
test.v10<- cbind(test[1:27],yt10)
for(i in 2:27)
 {
  test.v9[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
for(i in 2:27)
 {
  test.v10[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
```

```r
# scatter plot & cor
fitv10 <- lm(train$`V-10`~ ., data = train.v10)
summary(fitv10)
```

```
##
## Call:
## lm(formula = train$`V-10` ~ ., data = train.v10)
##
## Residuals:
##     Min      1Q Median      3Q     Max
## -96.94 -11.10   0.13  11.22 195.69
##
```

```
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 235.4539     8.1451  28.907  < 2e-16 ***
## Zipcode2      3.2450     9.3795   0.346  0.72962    
## Zipcode3     -9.8061     8.1755  -1.199  0.23134    
## Zipcode4     -5.4482     9.8506  -0.553  0.58064    
## Zipcode5     -0.7110    10.4575  -0.068  0.94584    
## Zipcode6     -4.8886     9.8547  -0.496  0.62023    
## Zipcode7    -12.5746    10.3617  -1.214  0.22591    
## Zipcode8     -5.6590    10.5311  -0.537  0.59144    
## Zipcode9     -7.8606    19.2677  -0.408  0.68360    
## Zipcode10   -16.8070    16.3271  -1.029  0.30416    
## Zipcode11   -14.2515    16.6566  -0.856  0.39293    
## Zipcode12   -14.8322    12.3713  -1.199  0.23154    
## Zipcode13   -17.3230    12.8140  -1.352  0.17747    
## Zipcode14   -14.0094    11.3208  -1.237  0.21691    
## Zipcode15   -26.3242    14.0832  -1.869  0.06261 .  
## Zipcode16    -9.1743    14.7849  -0.621  0.53541    
## Zipcode17   -15.7814    12.0238  -1.313  0.19039    
## Zipcode18   -16.1880    12.8378  -1.261  0.20834    
## Zipcode19   -11.4882    12.6624  -0.907  0.36502    
## Zipcode20   -11.4787    11.4433  -1.003  0.31666    
## `V-2`         4.1560     7.5873   0.548  0.58429    
## `V-3`        -6.6246     5.9795  -1.108  0.26883    
## `V-4`         7.5558     3.7532   2.013  0.04503 *  
## `V-5`       150.2443     7.3503  20.441  < 2e-16 ***
## `V-6`        -7.3692     3.2394  -2.275  0.02365 *  
## `V-7`        29.0769     1.7009  17.095  < 2e-16 ***
## `V-8`        -5.8754     4.2381  -1.386  0.16672    
## `V-11`       -2.1577     3.9016  -0.553  0.58067    
## `V-12`       -8.6235    41.5064  -0.208  0.83556    
## `V-13`       17.5093    28.9961   0.604  0.54642    
## `V-14`        4.0457     3.8376   1.054  0.29267    
## `V-15`       63.3510    28.4569   2.226  0.02677 *  
## `V-16`       14.3414     8.5742   1.673  0.09549 .  
## `V-17`      -82.9451    15.5412  -5.337 1.92e-07 ***
## `V-18`      -11.6103     6.5866  -1.763  0.07901 .  
## `V-19`       -4.9097     7.3955  -0.664  0.50730    
## `V-20`       -0.8114     3.8823  -0.209  0.83460    
## `V-21`       -2.9399    16.5997  -0.177  0.85955    
## `V-22`      -55.8548    20.9688  -2.664  0.00816 ** 
## `V-23`       16.5962     7.3905   2.246  0.02549 *  
## `V-24`      -13.1186     8.9559  -1.465  0.14407    
## `V-25`       57.5260    78.4623   0.733  0.46405    
## `V-26`       16.6635    62.8677   0.265  0.79116    
## `V-27`      -11.0637     7.3525  -1.505  0.13348    
## `V-28`        8.9552     4.6603   1.922  0.05565 .  
## `V-29`       -7.1743    14.1937  -0.505  0.61362    
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 27.61 on 288 degrees of freedom
## Multiple R-squared:  0.9729, Adjusted R-squared:  0.9687 
## F-statistic: 229.7 on 45 and 288 DF,  p-value: < 2.2e-16
```

```r
cor(train.v10[,c(2:28)])
```

```
##              V-2          V-3         V-4         V-5         V-6
## V-2   1.000000000  0.945600958  0.77436762  0.23420508  0.21230859
## V-3   0.945600958  1.000000000  0.64019309  0.15528248  0.12979894
## V-4   0.774367618  0.640193087  1.00000000  0.57121240  0.33816723
## V-5   0.234205081  0.155282482  0.57121240  1.00000000  0.32722179
## V-6   0.212308594  0.129798937  0.33816723  0.32722179  1.00000000
## V-7   0.161547288  0.089225497  0.18086665  0.06782641  0.14136645
## V-8   0.226480811  0.145503571  0.47418223  0.80942131  0.16406550
## V-11 -0.035418005 -0.013619707  0.02290495  0.23425775 -0.06469386
## V-12  0.089149149  0.082405943  0.32119612  0.79164543 -0.19515942
## V-13  0.088261381  0.076359505  0.31999660  0.79468122 -0.18162136
## V-14 -0.031042870 -0.007498359  0.05373985  0.24842339 -0.06594751
## V-15  0.079502198  0.075962603  0.30494320  0.77237472 -0.20802750
## V-16  0.092661957  0.094008848  0.29144067  0.70758556 -0.17878957
## V-17  0.092216627  0.086045671  0.31811051  0.78139809 -0.18702009
## V-18  0.067722258  0.051402223  0.20398832  0.48986363 -0.05370173
## V-19  0.063634669  0.061394585  0.25013127  0.64261538 -0.16845274
## V-20 -0.007826318 -0.030715653 -0.04911039 -0.21740676  0.11616290
## V-21  0.083525605  0.073809085  0.31314787  0.78345004 -0.20180050
## V-22  0.100604965  0.092276694  0.32827391  0.78788394 -0.19693094
## V-23  0.094428570  0.062129772  0.31781905  0.72055691 -0.16593281
## V-24  0.082614838  0.062240681  0.26826007  0.64481091 -0.08200588
## V-25  0.089286179  0.073363554  0.32671357  0.79833830 -0.18004635
## V-26  0.090209687  0.074963521  0.32844952  0.79873355 -0.18638055
## V-27  0.098501142  0.066699364  0.32827781  0.72016802 -0.14959341
## V-28  0.081782289  0.065819345  0.15407731  0.30938116 -0.01506138
## V-29  0.079099698  0.074670140  0.30292846  0.78219642 -0.19173899
## V-10  0.263265794  0.166628000  0.59551162  0.96061711  0.31582150
##              V-7         V-8         V-11        V-12         V-13
## V-2   0.161547288  0.22648081 -0.035418005  0.08914915  0.088261381
## V-3   0.089225497  0.14550357 -0.013619707  0.08240594  0.076359505
## V-4   0.180866645  0.47418223  0.022904947  0.32119612  0.319996595
## V-5   0.067826410  0.80942131  0.234257753  0.79164543  0.794681216
## V-6   0.141366451  0.16406550 -0.064693859 -0.19515942 -0.181621357
## V-7   1.000000000  0.01776631  0.002002689 -0.01102056  0.001928748
## V-8   0.017766307  1.00000000  0.214791473  0.62970430  0.634816454
## V-11  0.002002689  0.21479147  1.000000000  0.31244811  0.344567028
## V-12 -0.011020564  0.62970430  0.312448115  1.00000000  0.990115542
## V-13  0.001928748  0.63481645  0.344567028  0.99011554  1.000000000
## V-14 -0.019721226  0.24596566  0.860365531  0.31564739  0.340203213
## V-15 -0.024051443  0.61691478  0.332852738  0.98696187  0.965649585
## V-16 -0.001528973  0.54631418  0.413873083  0.91047002  0.911852344
## V-17 -0.007634694  0.61852139  0.360611448  0.98089806  0.976602879
## V-18  0.103500897  0.41133991  0.234032892  0.51878542  0.526198919
## V-19 -0.006441351  0.54873449  0.299067350  0.80590755  0.775279122
## V-20  0.074979304 -0.13666349 -0.264273120 -0.35712077 -0.289603798
## V-21 -0.001781932  0.61813693  0.327402884  0.99252339  0.983217541
## V-22  0.005911640  0.61471306  0.307851986  0.99291719  0.980110935
## V-23  0.046973086  0.56744724  0.111823359  0.85524646  0.868914803
## V-24  0.066728819  0.51811642  0.409866559  0.74751228  0.810329992
## V-25  0.017701519  0.63912703  0.346748048  0.98283160  0.993948015
## V-26  0.012712520  0.64038848  0.328935009  0.98773368  0.992846711
```

```
## V-27  0.041148951  0.58009504  0.101545061  0.83897307  0.859104077
## V-28  0.097650836  0.27813186  0.104124516  0.29277178  0.286868723
## V-29 -0.020623125  0.63038431  0.302106965  0.98201195  0.970009704
## V-10  0.246356641  0.77602067  0.216265318  0.75362621  0.758385544
##              V-14        V-15         V-16         V-17        V-18
## V-2  -0.031042870  0.07950220  0.092661957  0.092216627  0.06772226
## V-3  -0.007498359  0.07596260  0.094008848  0.086045671  0.05140222
## V-4   0.053739850  0.30494320  0.291440666  0.318110514  0.20398832
## V-5   0.248423394  0.77237472  0.707585561  0.781398086  0.48986363
## V-6  -0.065947512 -0.20802750 -0.178789572 -0.187020090 -0.05370173
## V-7  -0.019721226 -0.02405144 -0.001528973 -0.007634694  0.10350090
## V-8   0.245965661  0.61691478  0.546314176  0.618521388  0.41133991
## V-11  0.860365531  0.33285274  0.413873083  0.360611448  0.23403289
## V-12  0.315647390  0.98696187  0.910470016  0.980898057  0.51878542
## V-13  0.340203213  0.96564959  0.911852344  0.976602879  0.52619892
## V-14  1.000000000  0.32141733  0.429650641  0.377683833  0.22975516
## V-15  0.321417327  1.00000000  0.891777513  0.965041224  0.53798776
## V-16  0.429650641  0.89177751  1.000000000  0.945888259  0.47463555
## V-17  0.377683833  0.96504122  0.945888259  1.000000000  0.51550667
## V-18  0.229755159  0.53798776  0.474635548  0.515506674  1.00000000
## V-19  0.280428299  0.85592540  0.710762676  0.763396891  0.76005321
## V-20 -0.223180633 -0.42477138 -0.460695728 -0.421048499 -0.04807152
## V-21  0.314238352  0.98479930  0.905593128  0.970529084  0.52651551
## V-22  0.308650562  0.98188751  0.922203466  0.977496291  0.54096321
## V-23  0.094598910  0.83560127  0.752719249  0.849345735  0.47191261
## V-24  0.385324013  0.67979203  0.660607493  0.721302456  0.51287317
## V-25  0.337258655  0.95920948  0.886957004  0.963501199  0.55244157
## V-26  0.321973307  0.96848328  0.892081494  0.969094040  0.54969275
## V-27  0.139606014  0.79419440  0.681317005  0.818461854  0.39502159
## V-28  0.169608370  0.30630060  0.229200249  0.291059956  0.86463576
## V-29  0.299937377  0.98315243  0.866570249  0.946223759  0.53018537
## V-10  0.217431999  0.74070323  0.657945916  0.731985430  0.47370439
##              V-19         V-20         V-21        V-22        V-23
## V-2   0.063634669 -0.007826318  0.083525605  0.10060497  0.09442857
## V-3   0.061394585 -0.030715653  0.073809085  0.09227669  0.06212977
## V-4   0.250131268 -0.049110392  0.313147874  0.32827391  0.31781905
## V-5   0.642615376 -0.217406759  0.783450035  0.78788394  0.72055691
## V-6  -0.168452740  0.116162896 -0.201800496 -0.19693094 -0.16593281
## V-7  -0.006441351  0.074979304 -0.001781932  0.00591164  0.04697309
## V-8   0.548734491 -0.136663489  0.618136933  0.61471306  0.56744724
## V-11  0.299067350 -0.264273120  0.327402884  0.30785199  0.11182336
## V-12  0.805907545 -0.357120769  0.992523387  0.99291719  0.85524646
## V-13  0.775279122 -0.289603798  0.983217541  0.98011094  0.86891480
## V-14  0.280428299 -0.223180633  0.314238352  0.30865056  0.09459891
## V-15  0.855925405 -0.424771384  0.984799301  0.98188751  0.83560127
## V-16  0.710762676 -0.460695728  0.905593128  0.92220347  0.75271925
## V-17  0.763396891 -0.421048499  0.970529084  0.97749629  0.84934574
## V-18  0.760053214 -0.048071518  0.526515511  0.54096321  0.47191261
## V-19  1.000000000 -0.277789904  0.817092309  0.81052043  0.63333981
## V-20 -0.277789904  1.000000000 -0.359643069 -0.36626055 -0.22614493
## V-21  0.817092309 -0.359643069  1.000000000  0.99053656  0.84433800
## V-22  0.810520432 -0.366260547  0.990536564  1.00000000  0.84869783
## V-23  0.633339813 -0.226144932  0.844337998  0.84869783  1.00000000
## V-24  0.531494188  0.097835932  0.745168073  0.73102304  0.65975292
```

```
## V-25  0.781298186 -0.248323465  0.977863958  0.97321585  0.88823633
## V-26  0.792637630 -0.283334553  0.981947612  0.97801582  0.89700001
## V-27  0.570548168 -0.069144581  0.824139284  0.81998426  0.91703848
## V-28  0.561385894  0.037568332  0.296720859  0.31415987  0.25026572
## V-29  0.828660567 -0.330418745  0.975786332  0.97155143  0.83140014
## V-10  0.625177036 -0.188652528  0.748987995  0.74845738  0.72261514
##             V-24        V-25        V-26        V-27        V-28
## V-2   0.08261484  0.08928618  0.09020969  0.09850114  0.08178229
## V-3   0.06224068  0.07336355  0.07496352  0.06669936  0.06581934
## V-4   0.26826007  0.32671357  0.32844952  0.32827781  0.15407731
## V-5   0.64481091  0.79833830  0.79873355  0.72016802  0.30938116
## V-6  -0.08200588 -0.18004635 -0.18638055 -0.14959341 -0.01506138
## V-7   0.06672882  0.01770152  0.01271252  0.04114895  0.09765084
## V-8   0.51811642  0.63912703  0.64038848  0.58009504  0.27813186
## V-11  0.40986656  0.34674805  0.32893501  0.10154506  0.10412452
## V-12  0.74751228  0.98283160  0.98773368  0.83897307  0.29277178
## V-13  0.81032999  0.99394802  0.99284671  0.85910408  0.28686872
## V-14  0.38532401  0.33725865  0.32197331  0.13960601  0.16960837
## V-15  0.67979203  0.95920948  0.96848328  0.79419440  0.30630060
## V-16  0.66060749  0.88695700  0.89208149  0.68131701  0.22920025
## V-17  0.72130246  0.96350120  0.96909404  0.81846185  0.29105996
## V-18  0.51287317  0.55244157  0.54969275  0.39502159  0.86463576
## V-19  0.53149419  0.78129819  0.79263763  0.57054817  0.56138589
## V-20  0.09783593 -0.24832347 -0.28333455 -0.06914458  0.03756833
## V-21  0.74516807  0.97786396  0.98194761  0.82413928  0.29672086
## V-22  0.73102304  0.97321585  0.97801582  0.81998426  0.31415987
## V-23  0.65975292  0.88823633  0.89700001  0.91703848  0.25026572
## V-24  1.00000000  0.83419437  0.80279114  0.70242848  0.25925649
## V-25  0.83419437  1.00000000  0.99785756  0.87754156  0.30716315
## V-26  0.80279114  0.99785756  1.00000000  0.88162945  0.31251731
## V-27  0.70242848  0.87754156  0.88162945  1.00000000  0.25408508
## V-28  0.25925649  0.30716315  0.31251731  0.25408508  1.00000000
## V-29  0.73163217  0.96492985  0.96786639  0.80227328  0.29412577
## V-10  0.61612424  0.76984469  0.77157245  0.71050664  0.30221239
##             V-29       V-10
## V-2   0.07909970  0.2632658
## V-3   0.07467014  0.1666280
## V-4   0.30292846  0.5955116
## V-5   0.78219642  0.9606171
## V-6  -0.19173899  0.3158215
## V-7  -0.02062312  0.2463566
## V-8   0.63038431  0.7760207
## V-11  0.30210697  0.2162653
## V-12  0.98201195  0.7536262
## V-13  0.97000970  0.7583855
## V-14  0.29993738  0.2174320
## V-15  0.98315243  0.7407032
## V-16  0.86657025  0.6579459
## V-17  0.94622376  0.7319854
## V-18  0.53018537  0.4737044
## V-19  0.82866057  0.6251770
## V-20 -0.33041874 -0.1886525
## V-21  0.97578633  0.7489880
## V-22  0.97155143  0.7484574
```

```
## V-23  0.83140014  0.7226151
## V-24  0.73163217  0.6161242
## V-25  0.96492985  0.7698447
## V-26  0.96786639  0.7715725
## V-27  0.80227328  0.7105066
## V-28  0.29412577  0.3022124
## V-29  1.00000000  0.7513661
## V-10  0.75136612  1.0000000
```

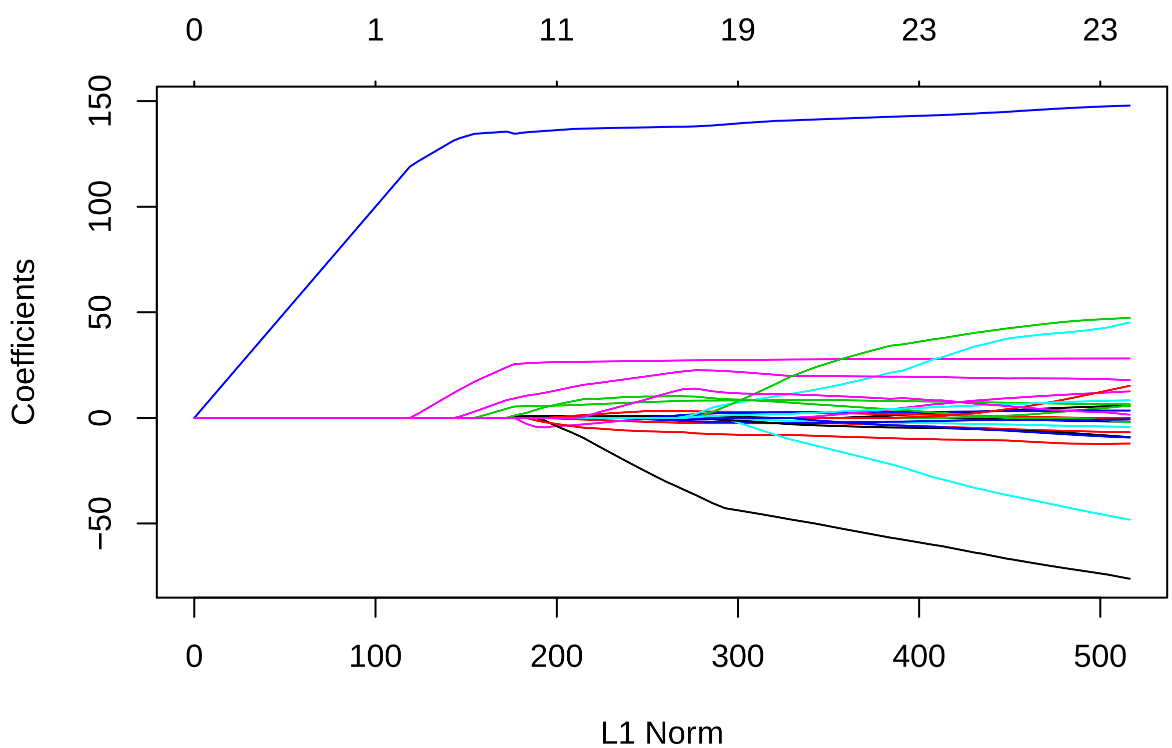
```
# Lasso to determine variable
library("glmnet")
```

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

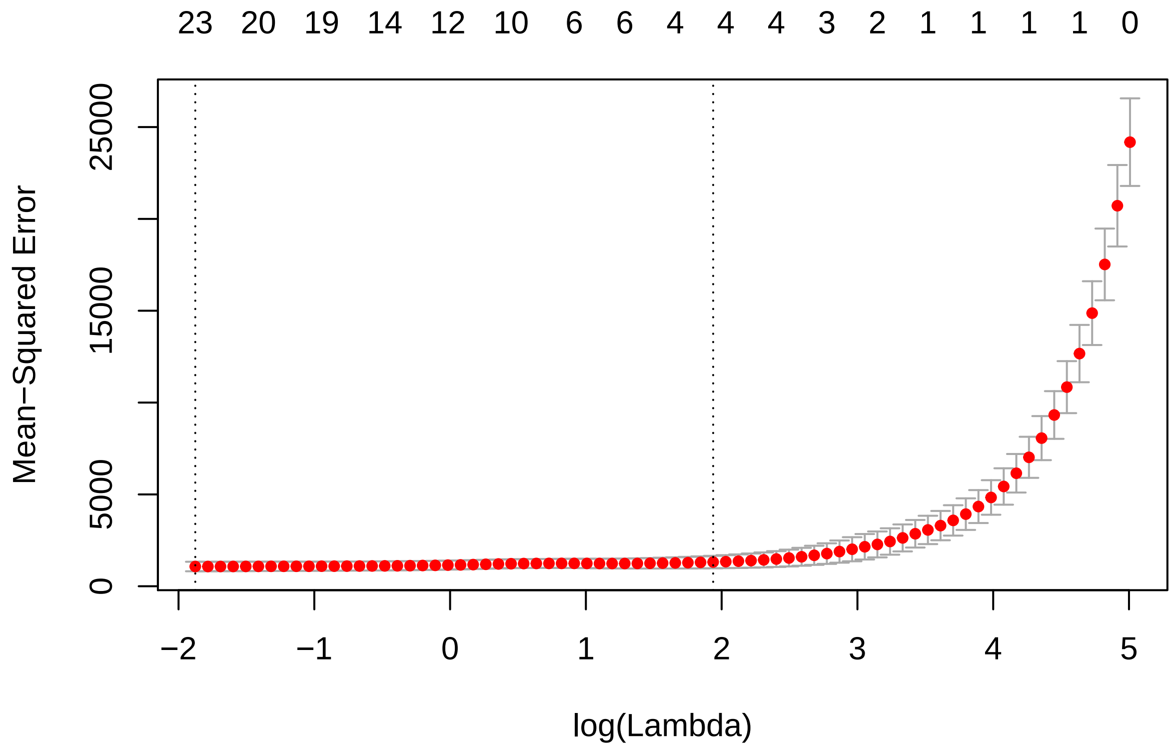
```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-16
```

```
x.simple=as.matrix(train.v10[,2:27])
y=train.v10$`V-10`
fitlasso=glmnet(x.simple,y,alpha = 1)
plot(fitlasso)
```



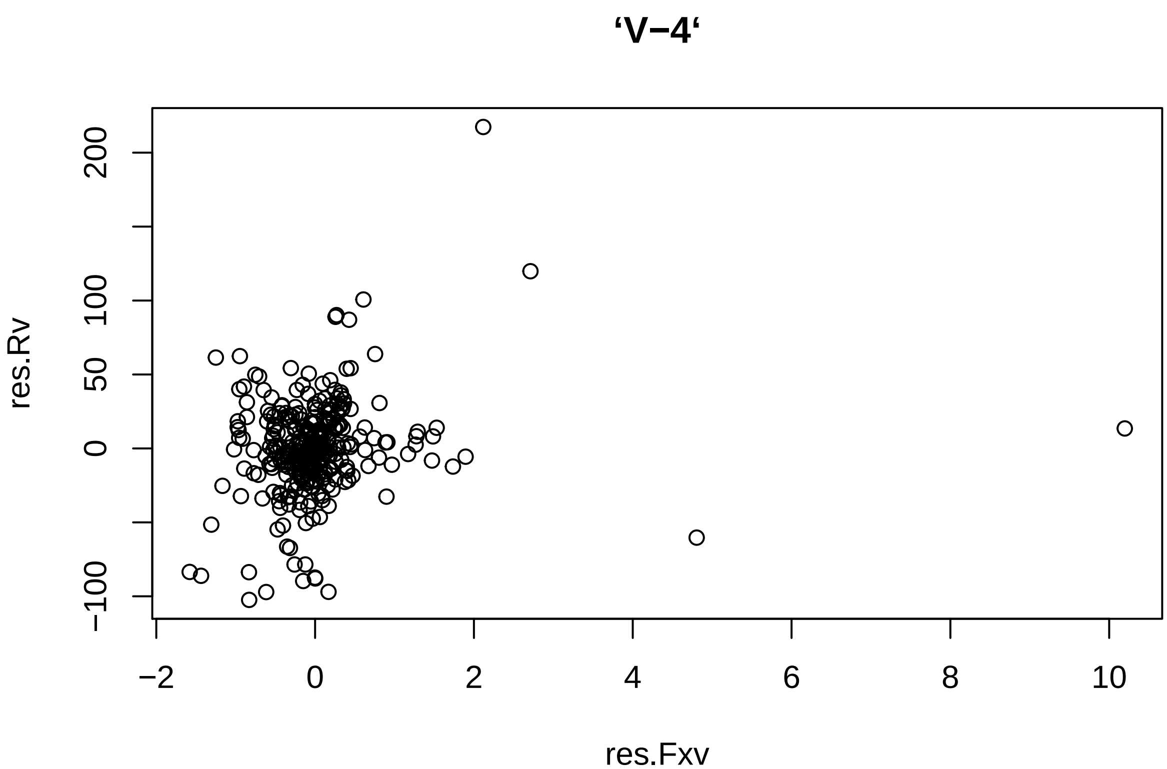

```
cv.lasso=cv.glmnet(x.simple,y)
plot(cv.lasso)
```



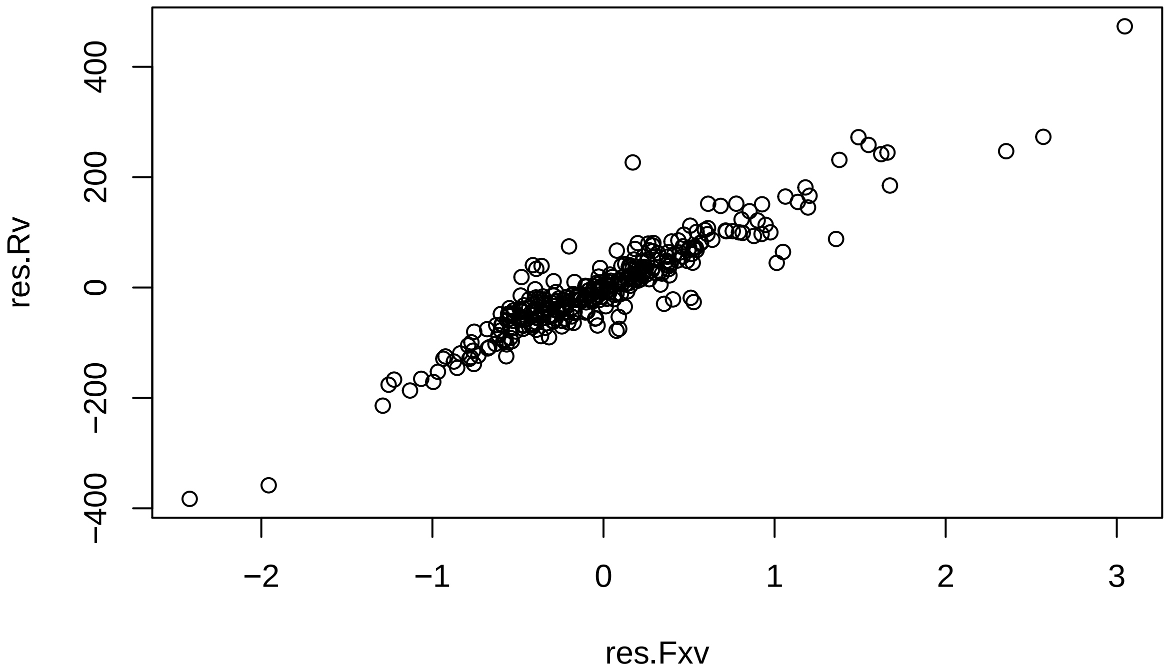

```r
coef(cv.lasso)
```

```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 226.392216
## V-2           .
## V-3           .
## V-4           2.749670
## V-5         135.175905
## V-6           .
## V-7          21.494386
## V-8           .
## V-11          .
## V-12          .
## V-13          .
## V-14          .
## V-15          .
## V-16          .
## V-17          .
## V-18          .
## V-19          .
## V-20          .
## V-21          .
## V-22          .
## V-23          6.441003
## V-24          .
## V-25          .
## V-26          .
## V-27          .
## V-28          .
## V-29          .
```

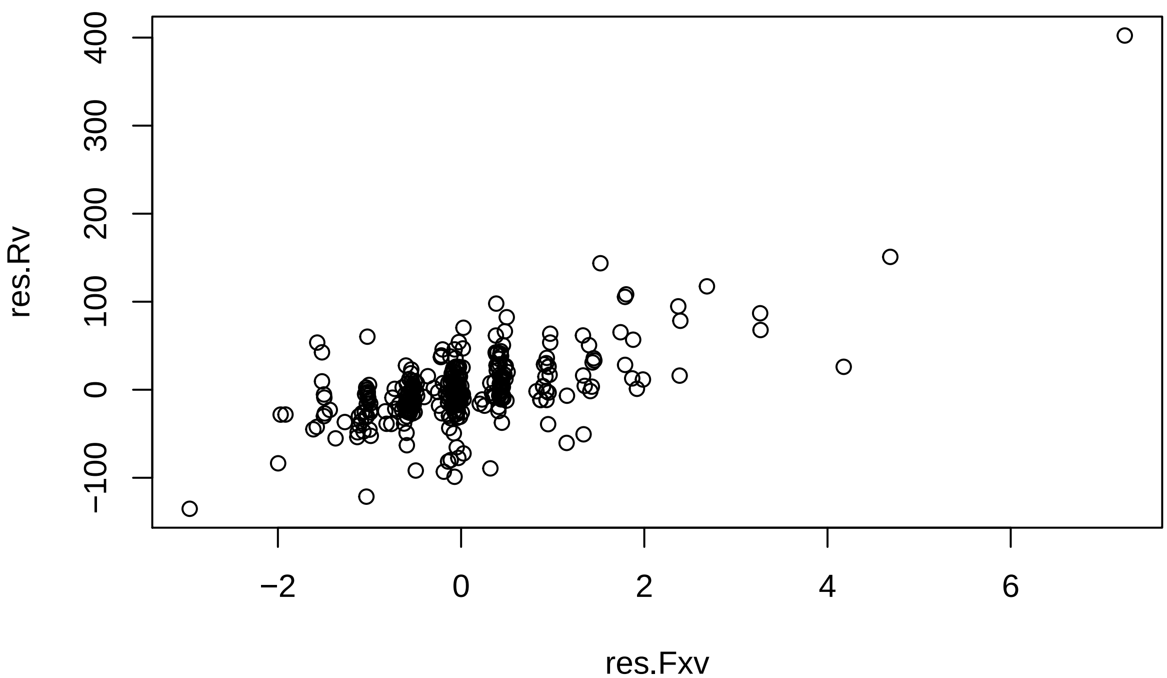
```r
# Model 3
# added variable factor to determine ^
datamodel3=data.frame(train.v10[,c(4,5,7,21,28)])
fitmodel3=lm(datamodel3$V.10~.,data = datamodel3)
colData3 <- list("`V-4`", "`V-5`", "`V-7`", "`V-23`")
names(colData3) <- c("`V-4`", "`V-5`", "`V-7`", "`V-23`")
removeXList <- colData3

for (rmX in removeXList){
  tmpV <- colData3
  tmpV[[rmX]] = NULL
  test.Rv=lm(as.formula(paste("`V-10` ~", paste(tmpV, collapse = "+"))), data = train.v10)
  res.Rv= test.Rv$residuals

  test.Fxv=lm(as.formula(paste(paste(rmX," ~"), paste(tmpV, collapse = "+"))), data = train.v10)
  res.Fxv= test.Fxv$residuals

  plot(res.Fxv,res.Rv,main = rmX)
}
```

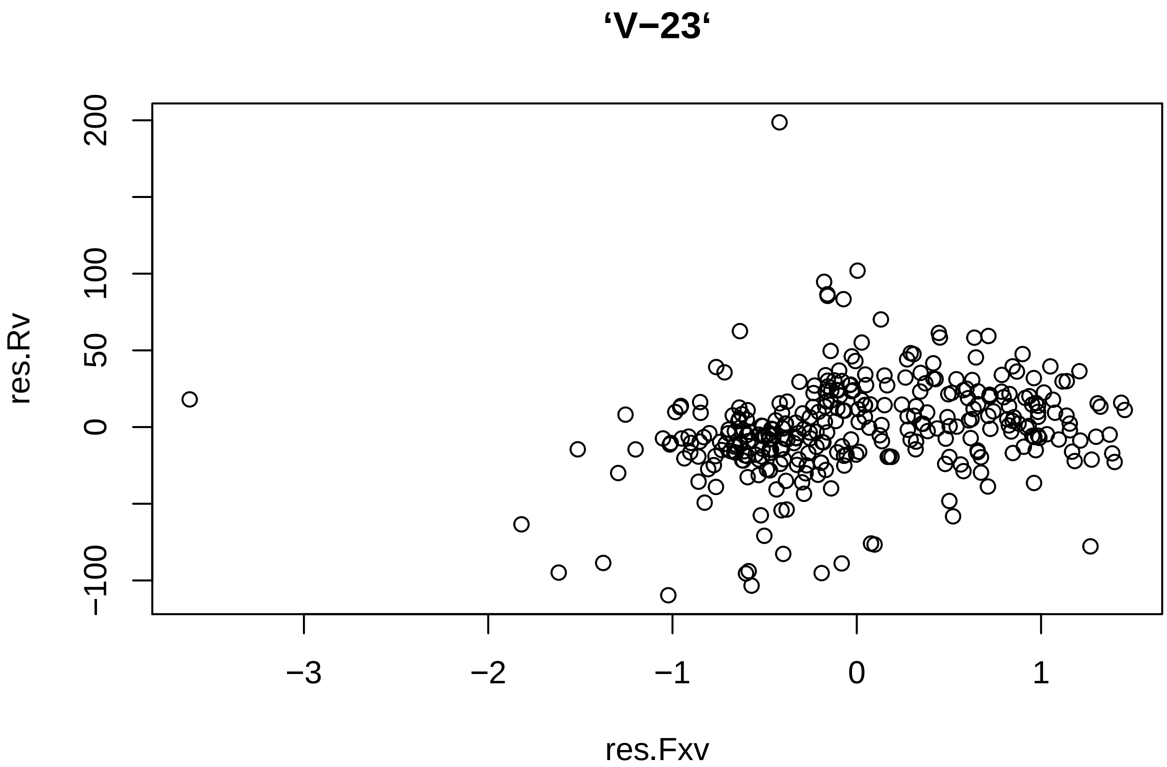


‘V−5‘


res.Rv
400
200
0
−200
−400
−2
−1
0
1
2
3
res.Fxv

‘V−7‘


res.Rv
400
300
200
100
0
−100
−2
0
2
4
6
res.Fxv




```r
summary(fitmodel3)$r.squared
```

```
## [1] 0.9588816
```

```r
summary(fitmodel3)$adj.r.squared
```

```
## [1] 0.9583816
```

```r
#Brown test whether constant variance and transformation for Model 3
resmodel3=fitmodel3$residuals
mmodel3=mean(datamodel3$V.10)
nmodel3=dim(datamodel3)[1]
p1=5
#1. Break the residuals into two groups.
Group1 <- resmodel3[datamodel3$V.10<mmodel3]
Group2 <-resmodel3[datamodel3$V.10>=mmodel3]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel3-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
 t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
 t
```

```
## [1] -2.798773
```

```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel3-p1-1)   # find the catical value
```

```
## [1] 1.967223
```

```r
# Weighted tranformation for model 3
wts <- 1/fitted(lm(abs(residuals(fitmodel3)) ~ ., data = datamodel3))^2

fitmodel3weight <- lm(datamodel3$V.10~ .,data = datamodel3, weights=wts)
datamodel3weight=cbind(datamodel3[1:4],datamodel3$V.10*wts)
summary(fitmodel3weight)$r.squared
```

```
## [1] 0.9615326
```

```r
summary(fitmodel3weight)$adj.r.squared
```

```
## [1] 0.961065
```

```r
#Brown test whether constant variance and transformation for Model 3 after tranformation
resmodel3b=fitmodel3weight $residuals
mmodel3=mean(datamodel3weight$`datamodel3$V.10 * wts`)
nmodel3=dim(datamodel3weight)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel3b[datamodel3weight$`datamodel3$V.10 * wts`<mmodel3]
Group2 <-resmodel3b[datamodel3weight$`datamodel3$V.10 * wts`>=mmodel3]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel3-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
t
```

```
## [1] 0.3504877
```

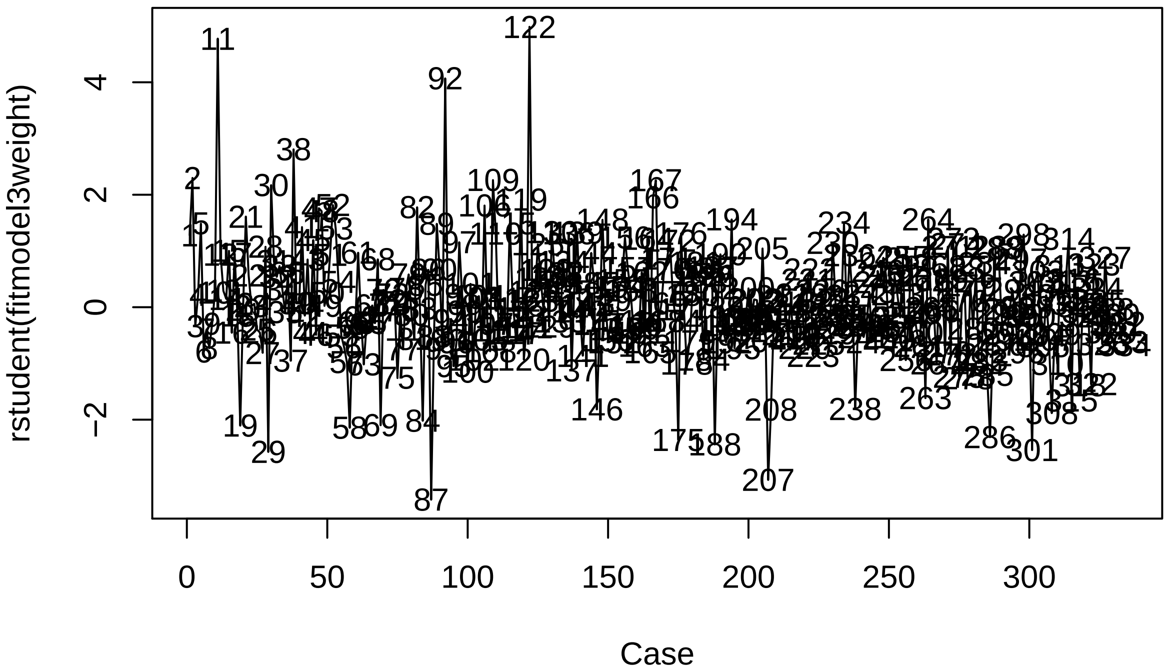
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel3-p1-1)   # find the catical value
```

```
## [1] 1.967223
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel3-p1-1))
```

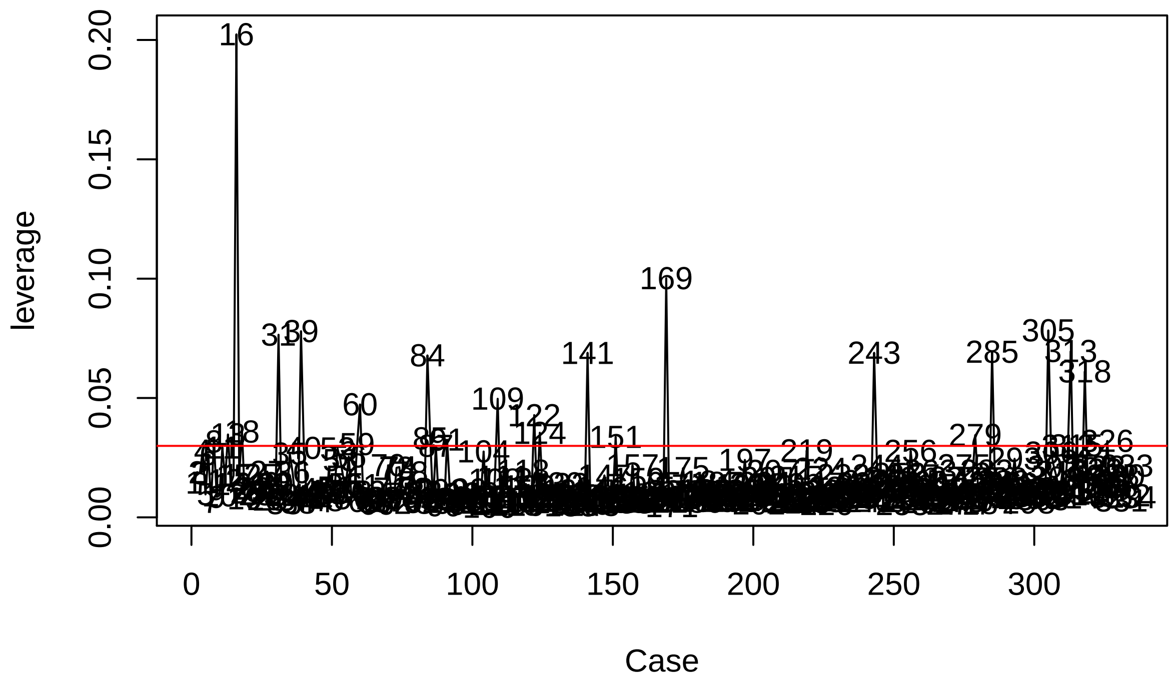
```
## [1] 0.7261977
```

```r
#y outlier for model3
Case <- c(1:nmodel3)
plot(Case, rstudent(fitmodel3weight), type="l")
text(Case, rstudent(fitmodel3weight), Case)
```




```r
alpha <- 0.05
crit <- qt(1-alpha/2/nmodel3, nmodel3-p1-1)
youtlier3=which(abs(rstudent(fitmodel3weight)) >=crit )
```

```r
#x outlier for model3
X <- as.matrix(cbind(rep(1,nmodel3), datamodel3[1:4]))
H <- X%*%solve(t(X)%*%X, tol=1e-20)%*%t(X)
leverage <- hatvalues(fitmodel3weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p1/nmodel3, col=2)
```




```r
xoutlier1=data.frame(which(leverage>2*p1/nmodel3) )
xoutlier1
```

```
##      which.leverage...2...p1.nmodel3.
## 8                                   8
## 13                                 13
## 16                                 16
## 18                                 18
## 31                                 31
## 39                                 39
## 59                                 59
## 60                                 60
## 84                                 84
## 85                                 85
## 87                                 87
## 91                                 91
## 109                               109
## 122                               122
## 124                               124
## 141                               141
## 151                               151
## 169                               169
## 243                               243
## 279                               279
## 285                               285
## 305                               305
## 312                               312
## 313                               313
## 315                               315
## 318                               318
## 326                               326
```

```r
#test whether outlier in the extend of the model3
IM3=influence.measures(fitmodel3weight)
dxoutlier3=union(which(IM3$infmat[,8]>0.2),which(IM3$infmat[,6]>2*sqrt(p1/nmodel3)))
#combine x and y outlier
finaloutlier3=union(dxoutlier3,youtlier3)
datamodel3Final=datamodel3[-c(finaloutlier3),]
# get model1 without x y outlier
fitmodel3x1=lm(datamodel3Final$V.10~.,data = datamodel3Final)
wtsx3 <- 1/fitted(lm(abs(residuals(fitmodel3x1)) ~ ., data = datamodel3Final))^2
Fmodel3=lm(datamodel3Final$V.10~., data = datamodel3Final,weights =wtsx3)
# R2 & adj R2 for model3 test
summary(Fmodel3)$r.squared
```

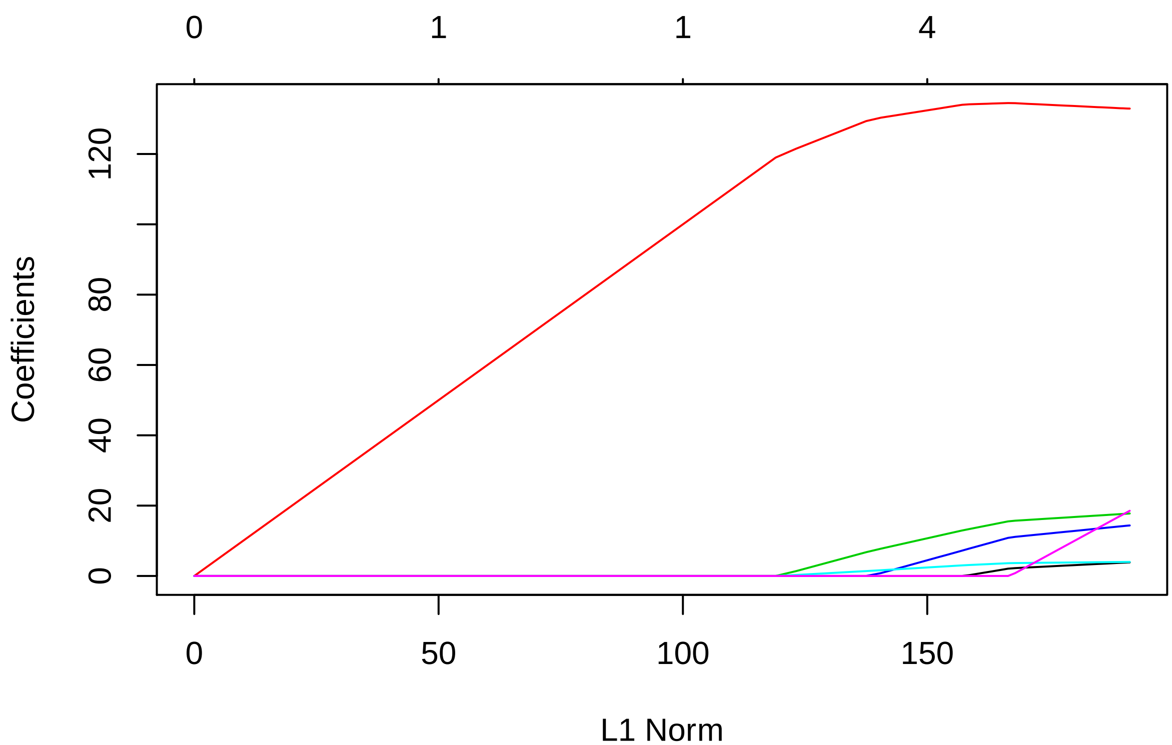
```
## [1] 0.9768281
```

```r
summary(Fmodel3)$adj.r.squared
```

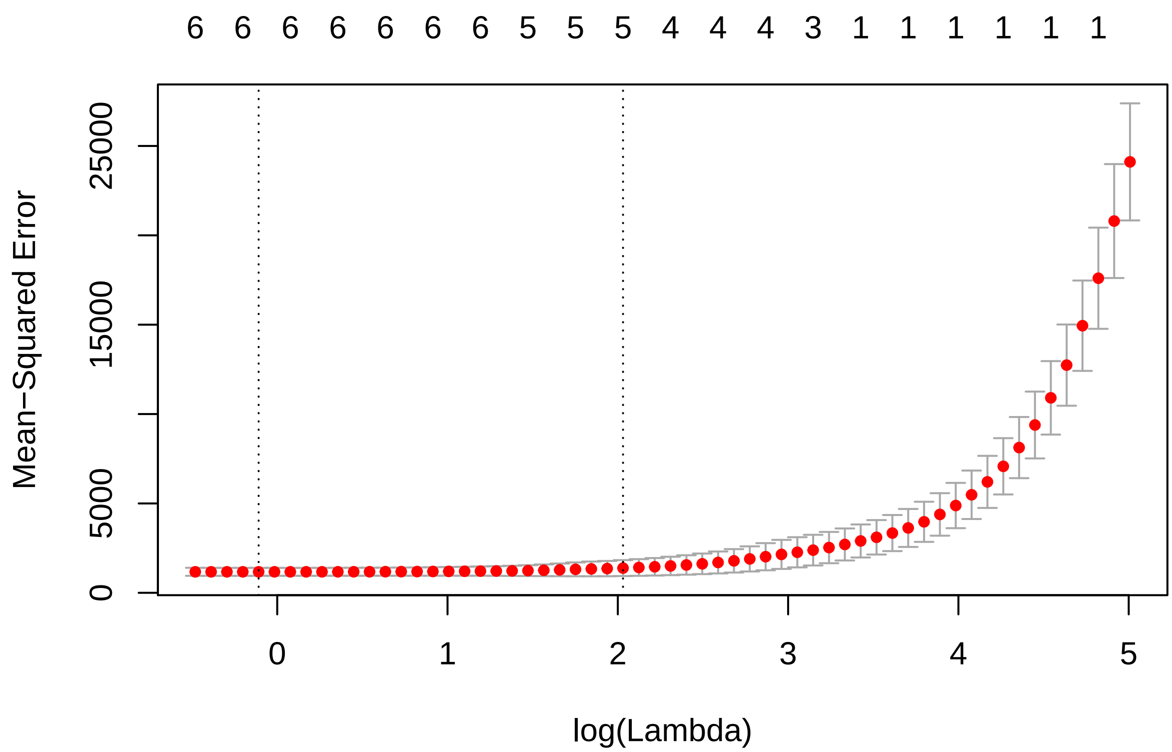
```
## [1] 0.9765403
```

```r
# add ^2 for model4
Data.new3 <- cbind(train.v10$`V-4`, train.v10$`V-5`, train.v10$`V-7`, train.v10$ `V-23`)
x3.new=as.matrix(cbind(Data.new3,((Data.new3)^2)[,-2]))
colnames(x3.new)=c("V-4","V-5","V-7","V-23","V-4.2","V-7.2","V-23.2")
```

```r
#lasso test x^2
library("glmnet")
fitlasso.x3add=glmnet(x3.new,y,alpha = 1)
plot(fitlasso.x3add)
```




```r
cv.lasso.x3add=cv.glmnet(x3.new,y)
plot(cv.lasso.x3add)
```




```
coef(cv.lasso.x3add)
```

```
## 8 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) 223.2886614
## V-4           0.2221622
## V-5         134.1264851
## V-7          13.3285514
## V-23          7.7699144
## V-4.2         .
## V-7.2         3.1128741
## V-23.2        .
```

```
# Model 4
trainv14 = data.frame(x3.new,y)
datamodel4=data.frame(trainv14[,c(2,8)])
fitmodel4=lm(datamodel4$y~.,data = datamodel4)
summary(fitmodel4)$r.squared
```

```
## [1] 0.9227852
```

```
summary(fitmodel4)$adj.r.squared
```

```
## [1] 0.9225527
```

```
#Brown test whether constant variance and transformation for Model 4
fitmodel4=lm(datamodel4$y~.,data = datamodel4)
resmodel4=fitmodel4$residuals
mmodel4=mean(datamodel4$y)
nmodel4=dim(datamodel4)[1]
#1. Break the residuals into two groups.
Group3 <- resmodel4[datamodel4$y<mmodel4]
Group4 <-resmodel4[datamodel4$y>=mmodel4]
```

```r
#2. Obtain the median of each group, using the commands:
M3 <- median(Group3)
M4 <- median(Group4)

#3. Obtain the mean absolute deviation for each group, using the commands:
D3 <- sum( abs( Group3 - M3 )) / length(Group3)
D4 <- sum( abs( Group4 - M4 )) / length(Group4)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group3 - M3) - D3 )^2 ) + sum( ( abs(Group4 - M4) - D4 )^2 ) ) / (nmodel4-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
 t <- ( D3 - D4 ) / ( s * sqrt( 1/length(Group3) + 1/length(Group4) ) )
 t
```

```
## [1] -5.216066
```

```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel4-p1-1)   # find the catical value
```

```
## [1] 1.967223
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel4-p1-1))
```

```
## [1] 3.244961e-07
```

```r
# Weighted tranformation for model 4
wts <- 1/fitted(lm(abs(residuals(fitmodel4)) ~ ., data = datamodel4))^2

fitmodel4weight <- lm(datamodel4$y~ .,data = datamodel4, weights=wts)
datamodel4weight=cbind(datamodel4[1],datamodel4$y*wts)
summary(fitmodel4weight)$r.squared
```

```
## [1] 0.8608815
```

```r
summary(fitmodel4weight)$adj.r.squared
```

```
## [1] 0.8604625
```

```r
#Brown test whether constant variance and transformation for Model 2 after tranformation
resmode22b=fitmodel4weight$residuals
mmodel4=mean(datamodel4weight$`datamodel4$y * wts`)
nmodel4=dim(datamodel4weight)[1]
#1. Break the residuals into two groups.
Group6 <- resmode22b[datamodel4weight$`datamodel4$y * wts`<mmodel4]
Group7 <- resmode22b[datamodel4weight$`datamodel4$y * wts`>=mmodel4]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group6)
M2 <- median(Group7)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group6 - M1 )) / length(Group6)
D2 <- sum( abs( Group7 - M2 )) / length(Group7)
```

```r
#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group6 - M1) - D1 )^2 ) + sum( ( abs(Group7 - M2) - D2 )^2 ) ) / (nmodel4-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group6) + 1/length(Group7) ) )
t
```

```
## [1] 1.5684
```

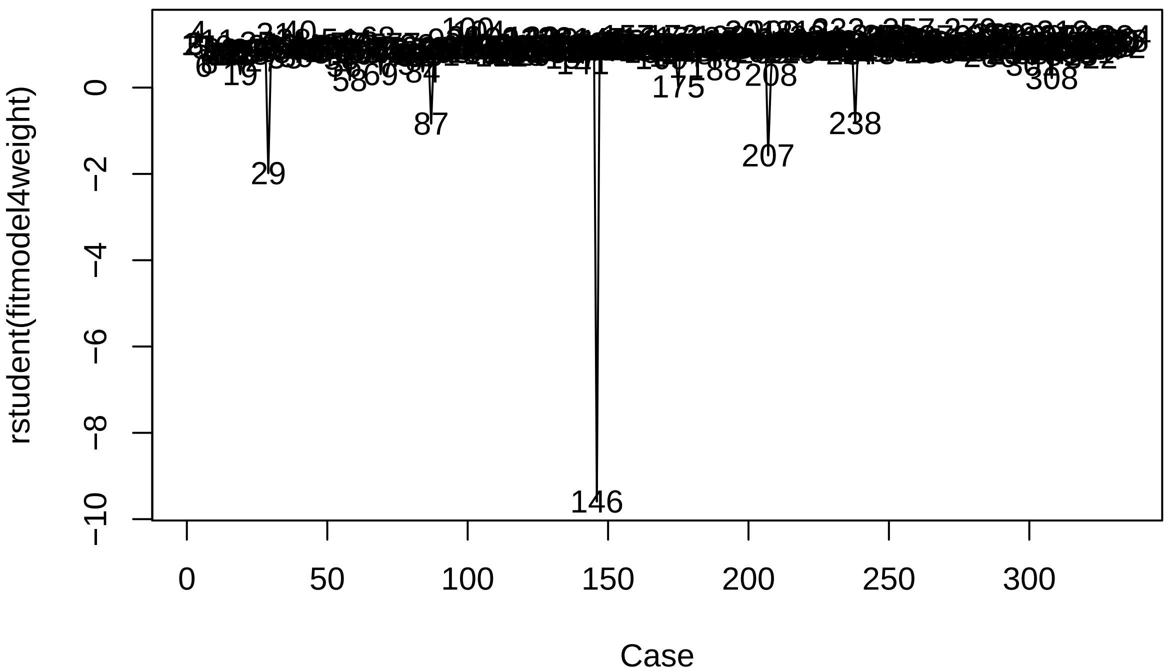
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel4-5)   # find the catical value
```

```
## [1] 1.967201
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel4-5))
```

```
## [1] 0.1177491
```

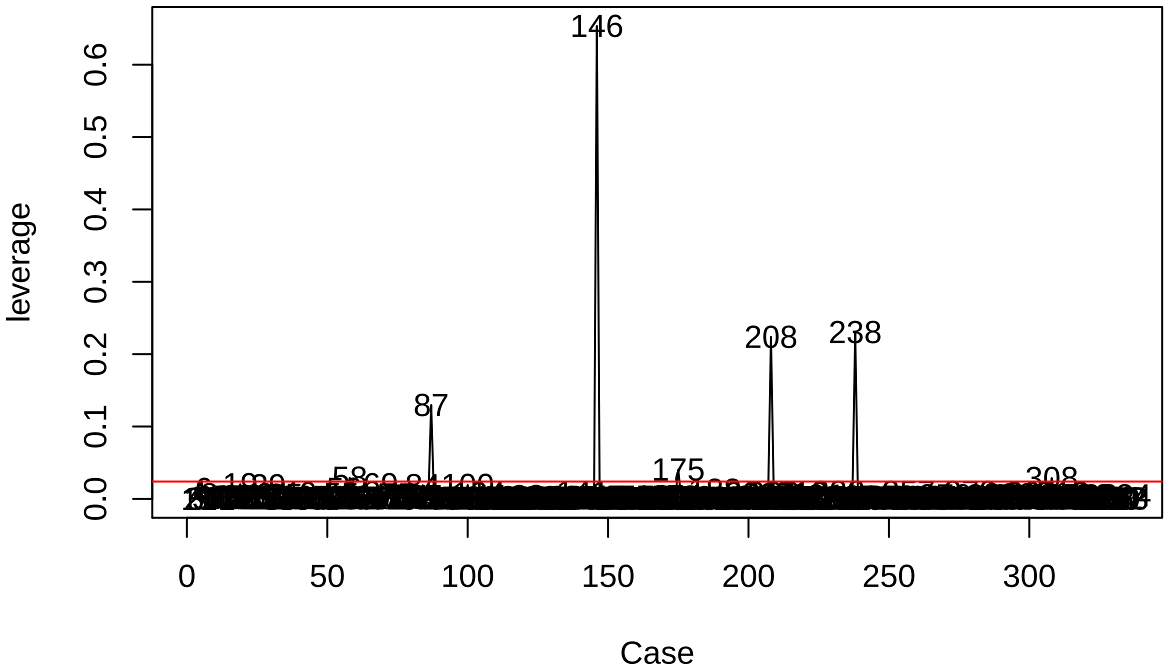
```r
#y outlier
Case <- c(1:nmodel4)
plot(Case, rstudent(fitmodel4weight), type="l")
text(Case, rstudent(fitmodel4weight), Case)
```




```r
alpha <- 0.01
p=4
crit <- qt(1-alpha/2/nmodel4, nmodel4-p-1)
youtlier=which(abs(rstudent(fitmodel4weight)) >=crit )
```

```r
#x outlier
X <- as.matrix(cbind(rep(1,nmodel4), datamodel4weight[1]))
H <- X%*%solve(t(X)%*%X,tol=1e-30)%*%t(X)
leverage <- hatvalues(fitmodel4weight)
```

```r
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p/nmodel4, col=2)
```




```r
xoutlier=data.frame(which(leverage>2*p/nmodel4) )
xoutlier
```

```
##     which.leverage...2...p.nmodel4.
## 58                               58
## 87                               87
## 146                             146
## 175                             175
## 208                             208
## 238                             238
## 308                             308
```

```r
#test whether outlier in the extend of the model
IM4=influence.measures(fitmodel4weight)
dxoutlier=union(which(IM4$infmat[,5]>0.2),which(IM4$infmat[,3]>2*sqrt(p/nmodel4)))
#combine x and y outlier
finaloutlier=union(dxoutlier,youtlier)
datamodel4Final=datamodel4[-c(finaloutlier),]
# get model2 without x y outlier
fitmodel4x2=lm(datamodel4Final$y~.,data = datamodel4Final)
wtsx2 <- 1/fitted(lm(abs(residuals(fitmodel4x2)) ~ ., data = datamodel4Final))^2
Fmodel4=lm(datamodel4Final$y~., data = datamodel4Final,weights =wtsx2)
# R2 & adj R2 for model1
summary(Fmodel4)$r.squared
```

```
## [1] 0.8804859
```

```r
summary(Fmodel4)$adj.r.squared
```

```
## [1] 0.8801249
```

```r
# Test the model
#strandized for test data
yt9=as.matrix(test$`V-9`)
colnames(yt9)=c("V-9")
yt10=as.matrix(test$`V-10`)
colnames(yt10)=c("V-10")
test.v9<- cbind(test[1:27],yt9)
test.v10<- cbind(test[1:27],yt10)
for(i in 2:27)
 {
  test.v9[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
for(i in 2:27)
 {
  test.v10[,i] <-(test[,i]-mean(test[,i])) /sd(test[,i])
}
```

```r
# scatter plot & cor
fitv10 <- lm(test$`V-10`~ ., data = test.v10)
summary(fitv10)
```

```
##
## Call:
## lm(formula = test$`V-10` ~ ., data = test.v10)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -10.870  -3.670   1.169   4.338  11.544
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  272.4457    11.3144  24.080 3.47e-10 ***
## `V-1`          1.5941     1.1706   1.362 0.203165
## `V-2`         -6.6147    24.4522  -0.271 0.792266
## `V-3`         40.1613    23.9268   1.679 0.124174
## `V-4`        -66.8744    22.6169  -2.957 0.014366 *
## `V-5`        233.8802    26.8405   8.714 5.53e-06 ***
## `V-6`         -6.9412    11.2577  -0.617 0.551292
## `V-7`         32.3178     3.2195  10.038 1.53e-06 ***
## `V-8`         30.6192     7.2639   4.215 0.001785 **
## `V-11`        -0.7395    14.8773  -0.050 0.961338
## `V-12`       113.1887   118.3086   0.957 0.361264
## `V-13`       140.3495    83.2022   1.687 0.122527
## `V-14`         1.3866    17.1699   0.081 0.937228
## `V-15`       169.7870    99.9927   1.698 0.120356
## `V-16`        10.0250    50.4989   0.199 0.846616
## `V-17`         9.1391    70.3806   0.130 0.899258
## `V-18`       -21.7666    18.2804  -1.191 0.261266
## `V-19`       -30.6981    21.3410  -1.438 0.180857
## `V-20`        77.3476    21.1456   3.658 0.004405 **
## `V-21`       -60.7189    42.0521  -1.444 0.179357
## `V-22`      -120.0430    66.2118  -1.813 0.099910 .
## `V-23`        73.7604    31.4409   2.346 0.040919 *
## `V-24`        11.8816    33.0547   0.359 0.726731
```

```
## `V-25`      -225.4046   268.3902  -0.840 0.420624
## `V-26`        66.3399   165.9609   0.400 0.697758
## `V-27`      -103.7999    22.2297  -4.669 0.000882 ***
## `V-28`        32.9507    15.8271   2.082 0.063992 .
## `V-29`       -19.0591    27.0856  -0.704 0.497699
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 11.78 on 10 degrees of freedom
## Multiple R-squared:  0.9991, Adjusted R-squared:  0.9968
## F-statistic:   423 on 27 and 10 DF,  p-value: 3.708e-12
```

```
cor(test.v10[,c(2:28)])
```

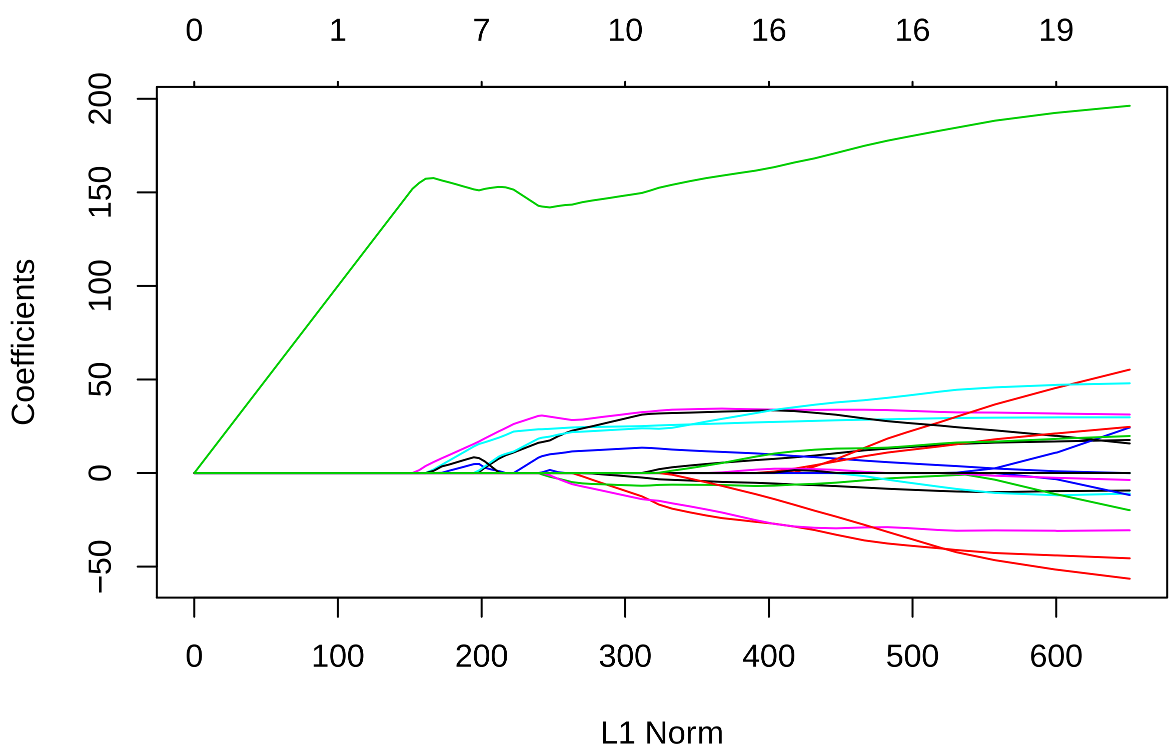
```
##              V-2         V-3         V-4         V-5         V-6
## V-2   1.00000000  0.98760515  0.73160086  0.37670831  0.22600652
## V-3   0.98760515  1.00000000  0.71209140  0.37228924  0.21567025
## V-4   0.73160086  0.71209140  1.00000000  0.84671012  0.24922766
## V-5   0.37670831  0.37228924  0.84671012  1.00000000  0.30351159
## V-6   0.22600652  0.21567025  0.24922766  0.30351159  1.00000000
## V-7  -0.02292599 -0.02186071  0.24412674  0.33620300  0.29322017
## V-8   0.53953700  0.53344562  0.86489429  0.82822710  0.30041366
## V-11  0.36533757  0.36353757  0.56934023  0.61657479  0.16658200
## V-12  0.33139692  0.31593503  0.73920733  0.86156959 -0.08532251
## V-13  0.32272403  0.30183748  0.73328812  0.85294637 -0.08492403
## V-14  0.28741154  0.26035082  0.55247646  0.64418155  0.26248596
## V-15  0.33136935  0.32684742  0.72021209  0.84528427 -0.09394580
## V-16  0.35132989  0.32660843  0.77856486  0.82834474 -0.03979719
## V-17  0.32506928  0.29793012  0.77490951  0.87052243 -0.05195761
## V-18 -0.13142833 -0.11399440  0.03433714  0.19025481 -0.24756714
## V-19  0.17836347  0.20812630  0.38801002  0.50391639 -0.18251913
## V-20 -0.29988016 -0.30073657 -0.69239049 -0.77660353 -0.05539371
## V-21  0.30817051  0.30110642  0.72947167  0.85704711 -0.10059098
## V-22  0.34513510  0.32903710  0.75979981  0.86558255 -0.08782856
## V-23  0.21719020  0.19529788  0.47802378  0.62976351 -0.21996723
## V-24  0.09327267  0.06689985  0.41470723  0.59337340 -0.12706560
## V-25  0.29584548  0.27747458  0.68387755  0.82511693 -0.12111300
## V-26  0.29535671  0.27861968  0.68366366  0.82505364 -0.13107651
## V-27  0.17990392  0.15830678  0.46120831  0.63685096 -0.13050909
## V-28 -0.22409534 -0.20295458 -0.13124054  0.04877237 -0.06067612
## V-29  0.35241519  0.33821433  0.74379471  0.84233751 -0.06839656
## V-10  0.37440101  0.37524116  0.81522709  0.97470602  0.31052398
##              V-7         V-8        V-11        V-12        V-13
## V-2  -0.02292599  0.53953700  0.36533757  0.33139692  0.32272403
## V-3  -0.02186071  0.53344562  0.36353757  0.31593503  0.30183748
## V-4   0.24412674  0.86489429  0.56934023  0.73920733  0.73328812
## V-5   0.33620300  0.82822710  0.61657479  0.86156959  0.85294637
## V-6   0.29322017  0.30041366  0.16658200 -0.08532251 -0.08492403
## V-7   1.00000000  0.30105455  0.11998472  0.22570889  0.23497926
## V-8   0.30105455  1.00000000  0.50374408  0.66459443  0.66941953
## V-11  0.11998472  0.50374408  1.00000000  0.57709819  0.58573941
## V-12  0.22570889  0.66459443  0.57709819  1.00000000  0.99284343
## V-13  0.23497926  0.66941953  0.58573941  0.99284343  1.00000000
## V-14  0.17104439  0.43539652  0.86567836  0.62321257  0.62133723
## V-15  0.24442844  0.65639165  0.60763465  0.98837433  0.97541628
```

```
## V-16  0.14701985  0.63479608  0.66093890  0.91934038  0.91825536
## V-17  0.22603833  0.65035736  0.60781001  0.97411641  0.97099506
## V-18  0.28047450 -0.04531235  0.16643626  0.35935593  0.36469287
## V-19  0.24933678  0.30909345  0.48169384  0.71663431  0.70108747
## V-20 -0.09126864 -0.60965008 -0.71249914 -0.75710367 -0.72547131
## V-21  0.24458981  0.67387434  0.61143932  0.98660922  0.97896018
## V-22  0.21226792  0.65935680  0.59549118  0.99072373  0.98163857
## V-23  0.21822375  0.46259694  0.38352358  0.79635337  0.82352013
## V-24  0.18430826  0.47278839  0.42093001  0.72829040  0.77718417
## V-25  0.24865855  0.63793373  0.56071173  0.98150465  0.99143887
## V-26  0.24245434  0.63248576  0.54894643  0.98406695  0.99057436
## V-27  0.29543427  0.44939056  0.22949717  0.77103403  0.79436596
## V-28  0.29970449 -0.19043096 -0.07412328  0.13729518  0.13530408
## V-29  0.23998363  0.66416219  0.57442695  0.98710408  0.97929565
## V-10  0.45747334  0.84754453  0.57655860  0.85482795  0.85117572
##               V-14       V-15        V-16        V-17        V-18
## V-2   0.2874115436  0.3313694  0.35132989  0.32506928 -0.13142833
## V-3   0.2603508218  0.3268474  0.32660843  0.29793012 -0.11399440
## V-4   0.5524764595  0.7202121  0.77856486  0.77490951  0.03433714
## V-5   0.6441815499  0.8452843  0.82834474  0.87052243  0.19025481
## V-6   0.2624859638 -0.0939458 -0.03979719 -0.05195761 -0.24756714
## V-7   0.1710443906  0.2444284  0.14701985  0.22603833  0.28047450
## V-8   0.4353965238  0.6563916  0.63479608  0.65035736 -0.04531235
## V-11  0.8656783621  0.6076346  0.66093890  0.60781001  0.16643626
## V-12  0.6232125688  0.9883743  0.91934038  0.97411641  0.35935593
## V-13  0.6213372286  0.9754163  0.91825536  0.97099506  0.36469287
## V-14  1.0000000000  0.6215299  0.72533932  0.69452099  0.15605408
## V-15  0.6215298730  1.0000000  0.90209227  0.94941667  0.40458450
## V-16  0.7253393223  0.9020923  1.00000000  0.96611641  0.27203608
## V-17  0.6945209943  0.9494167  0.96611641  1.00000000  0.31791123
## V-18  0.1560540827  0.4045845  0.27203608  0.31791123  1.00000000
## V-19  0.4083245027  0.7873891  0.59859493  0.62986553  0.73216288
## V-20 -0.7741418013 -0.7665163 -0.86187965 -0.80850985 -0.09154056
## V-21  0.6066935538  0.9906373  0.90719355  0.95103770  0.39962450
## V-22  0.6429662142  0.9786266  0.95341013  0.98380933  0.36255471
## V-23  0.4161109177  0.7774443  0.67084502  0.75610551  0.42932245
## V-24  0.4312455544  0.6872105  0.59685074  0.66803579  0.38651934
## V-25  0.5844609203  0.9675719  0.87566108  0.94225312  0.42548625
## V-26  0.5756684522  0.9713055  0.87711897  0.94478814  0.42811196
## V-27  0.3008578736  0.7255657  0.56500768  0.72278276  0.38691850
## V-28  0.0005952779  0.1613507  0.02801824  0.10393714  0.86355056
## V-29  0.5885690773  0.9831633  0.89956429  0.95135608  0.37296130
## V-10  0.5900117992  0.8516274  0.78108086  0.83674718  0.22666142
##            V-19        V-20       V-21        V-22       V-23        V-24
## V-2   0.1783635 -0.29988016  0.3081705  0.34513510  0.2171902  0.09327267
## V-3   0.2081263 -0.30073657  0.3011064  0.32903710  0.1952979  0.06689985
## V-4   0.3880100 -0.69239049  0.7294717  0.75979981  0.4780238  0.41470723
## V-5   0.5039164 -0.77660353  0.8570471  0.86558255  0.6297635  0.59337340
## V-6  -0.1825191 -0.05539371 -0.1005910 -0.08782856 -0.2199672 -0.12706560
## V-7   0.2493368 -0.09126864  0.2445898  0.21226792  0.2182237  0.18430826
## V-8   0.3090934 -0.60965008  0.6738743  0.65935680  0.4625969  0.47278839
## V-11  0.4816938 -0.71249914  0.6114393  0.59549118  0.3835236  0.42093001
## V-12  0.7166343 -0.75710367  0.9866092  0.99072373  0.7963534  0.72829040
## V-13  0.7010875 -0.72547131  0.9789602  0.98163857  0.8235201  0.77718417
```

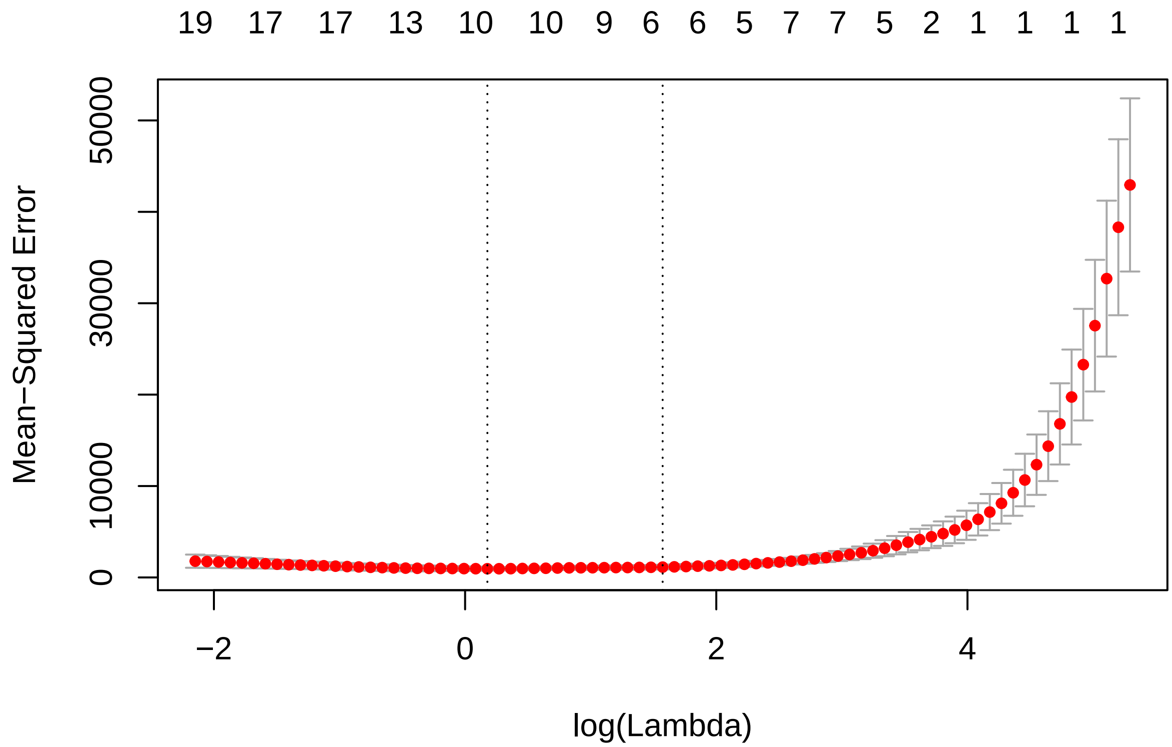
```
## V-14  0.4083245 -0.77414180  0.6066936  0.64296621  0.4161109  0.43124555
## V-15  0.7873891 -0.76651634  0.9906373  0.97862664  0.7774443  0.68721053
## V-16  0.5985949 -0.86187965  0.9071935  0.95341013  0.6708450  0.59685074
## V-17  0.6298655 -0.80850985  0.9510377  0.98380933  0.7561055  0.66803579
## V-18  0.7321629 -0.09154056  0.3996245  0.36255471  0.4293224  0.38651934
## V-19  1.0000000 -0.41602353  0.7641967  0.70151754  0.5605578  0.51266245
## V-20 -0.4160235  1.00000000 -0.7545829 -0.79468355 -0.4487491 -0.38553819
## V-21  0.7641967 -0.75458293  1.0000000  0.97974383  0.7731228  0.70924978
## V-22  0.7015175 -0.79468355  0.9797438  1.00000000  0.7662124  0.68447330
## V-23  0.5605578 -0.44874905  0.7731228  0.76621244  1.0000000  0.73658929
## V-24  0.5126625 -0.38553819  0.7092498  0.68447330  0.7365893  1.00000000
## V-25  0.7283520 -0.67082997  0.9695914  0.96181713  0.8723698  0.81802957
## V-26  0.7337861 -0.67297063  0.9715537  0.96501751  0.8762817  0.79843537
## V-27  0.4897397 -0.29678603  0.7330601  0.72350793  0.9020787  0.71872893
## V-28  0.5019291  0.13628274  0.1724028  0.13522069  0.2075548  0.13663264
## V-29  0.7447778 -0.73530060  0.9807783  0.97404091  0.7658337  0.71177550
## V-10  0.5596538 -0.70418735  0.8563429  0.84520316  0.6688126  0.61045344
##            V-25       V-26       V-27          V-28        V-29
## V-2   0.2958455  0.2953567  0.1799039 -0.2240953363  0.35241519
## V-3   0.2774746  0.2786197  0.1583068 -0.2029545779  0.33821433
## V-4   0.6838775  0.6836637  0.4612083 -0.1312405395  0.74379471
## V-5   0.8251169  0.8250536  0.6368510  0.0487723702  0.84233751
## V-6  -0.1211130 -0.1310765 -0.1305091 -0.0606761209 -0.06839656
## V-7   0.2486585  0.2424543  0.2954343  0.2997044861  0.23998363
## V-8   0.6379337  0.6324858  0.4493906 -0.1904309611  0.66416219
## V-11  0.5607117  0.5489464  0.2294972 -0.0741232850  0.57442695
## V-12  0.9815047  0.9840670  0.7710340  0.1372951846  0.98710408
## V-13  0.9914389  0.9905744  0.7943660  0.1353040806  0.97929565
## V-14  0.5844609  0.5756685  0.3008579  0.0005952779  0.58856908
## V-15  0.9675719  0.9713055  0.7255657  0.1613506662  0.98316326
## V-16  0.8756611  0.8771190  0.5650077  0.0280182377  0.89956429
## V-17  0.9422531  0.9447881  0.7227828  0.1039371380  0.95135608
## V-18  0.4254862  0.4281120  0.3869185  0.8635505633  0.37296130
## V-19  0.7283520  0.7337861  0.4897397  0.5019290952  0.74477779
## V-20 -0.6708300 -0.6729706 -0.2967860  0.1362827396 -0.73530060
## V-21  0.9695914  0.9715537  0.7330601  0.1724027847  0.98077831
## V-22  0.9618171  0.9650175  0.7235079  0.1352206913  0.97404091
## V-23  0.8723698  0.8762817  0.9020787  0.2075548473  0.76583369
## V-24  0.8180296  0.7984354  0.7187289  0.1366326393  0.71177550
## V-25  1.0000000  0.9989211  0.8352226  0.1848733515  0.96894426
## V-26  0.9989211  1.0000000  0.8395187  0.1902107103  0.97039969
## V-27  0.8352226  0.8395187  1.0000000  0.2995912846  0.73230796
## V-28  0.1848734  0.1902107  0.2995913  1.0000000000  0.14344307
## V-29  0.9689443  0.9703997  0.7323080  0.1434430723  1.00000000
## V-10  0.8365705  0.8355670  0.6675232  0.0919203396  0.84459157
##             V-10
## V-2   0.37440101
## V-3   0.37524116
## V-4   0.81522709
## V-5   0.97470602
## V-6   0.31052398
## V-7   0.45747334
## V-8   0.84754453
## V-11  0.57655860
```

```
## V-12  0.85482795
## V-13  0.85117572
## V-14  0.59001180
## V-15  0.85162743
## V-16  0.78108086
## V-17  0.83674718
## V-18  0.22666142
## V-19  0.55965376
## V-20 -0.70418735
## V-21  0.85634287
## V-22  0.84520316
## V-23  0.66881263
## V-24  0.61045344
## V-25  0.83657051
## V-26  0.83556696
## V-27  0.66752317
## V-28  0.09192034
## V-29  0.84459157
## V-10  1.00000000
```

```
# Lasso to determine variable
library("glmnet")
x.simple=as.matrix(test.v10[,2:27])
y=test.v10$`V-10`
fitlasso=glmnet(x.simple,y,alpha = 1)
plot(fitlasso)
```



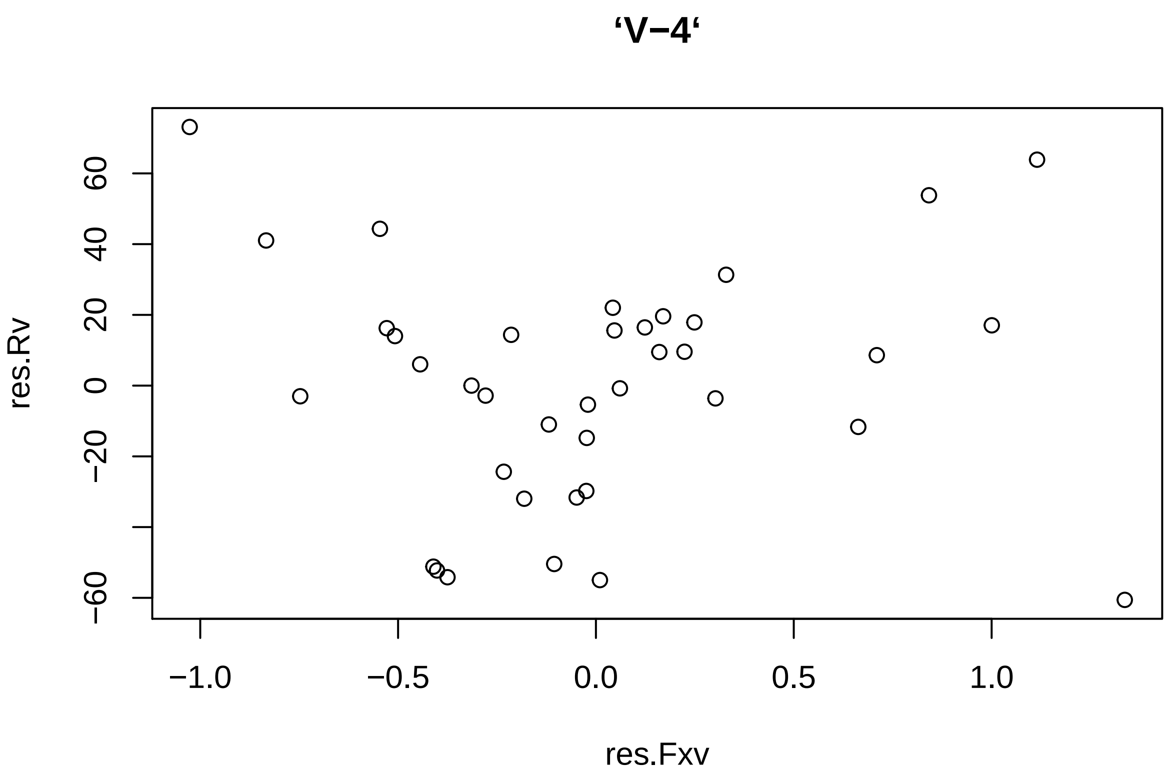

```
cv.lasso=cv.glmnet(x.simple,y)
plot(cv.lasso)
```

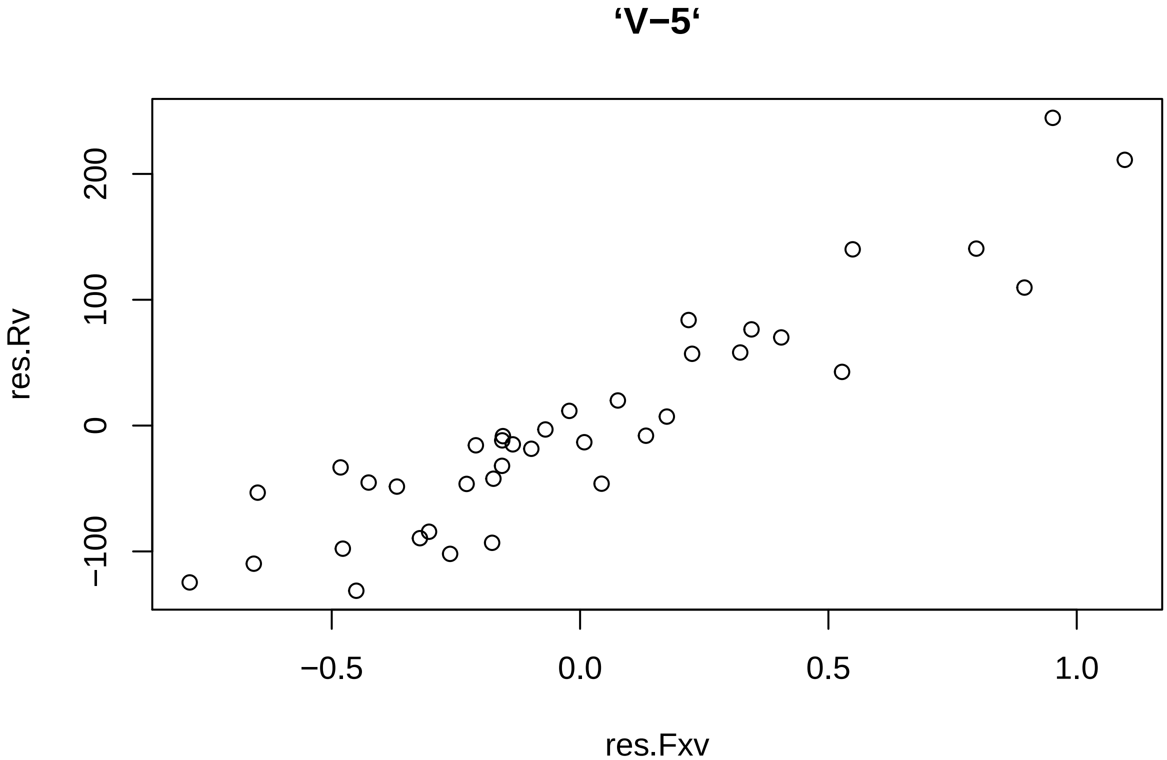


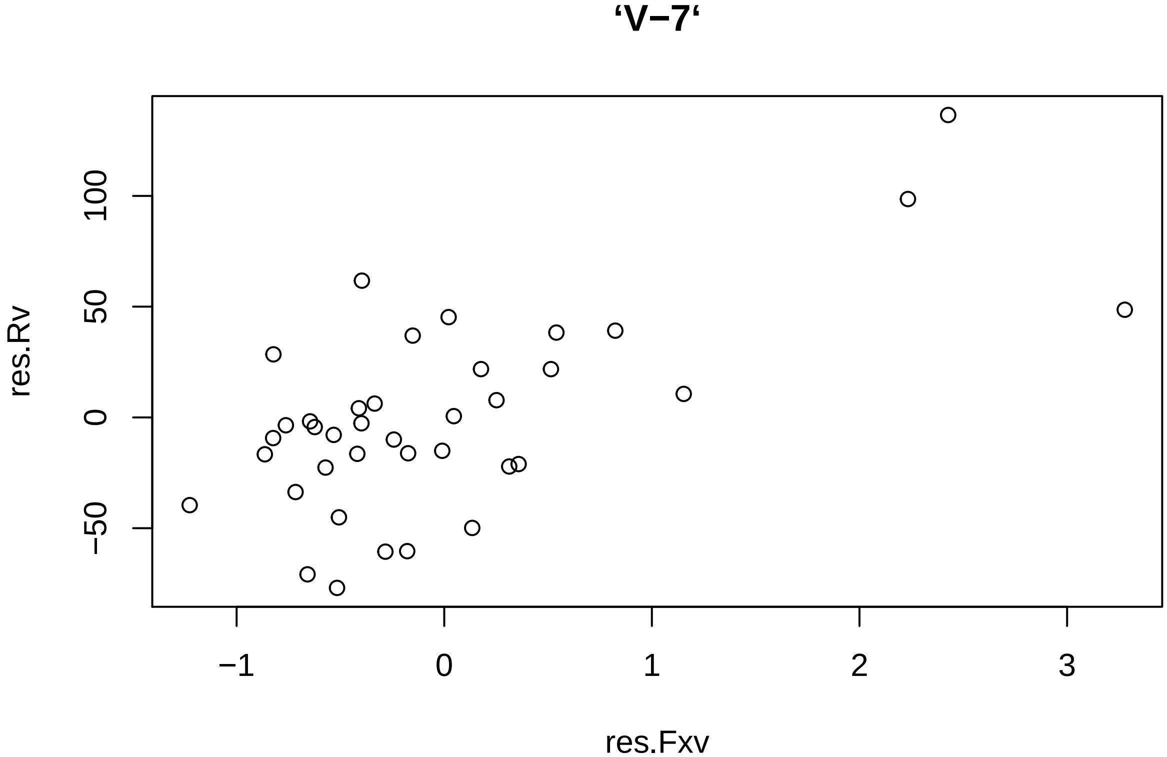

```r
coef(cv.lasso)
```

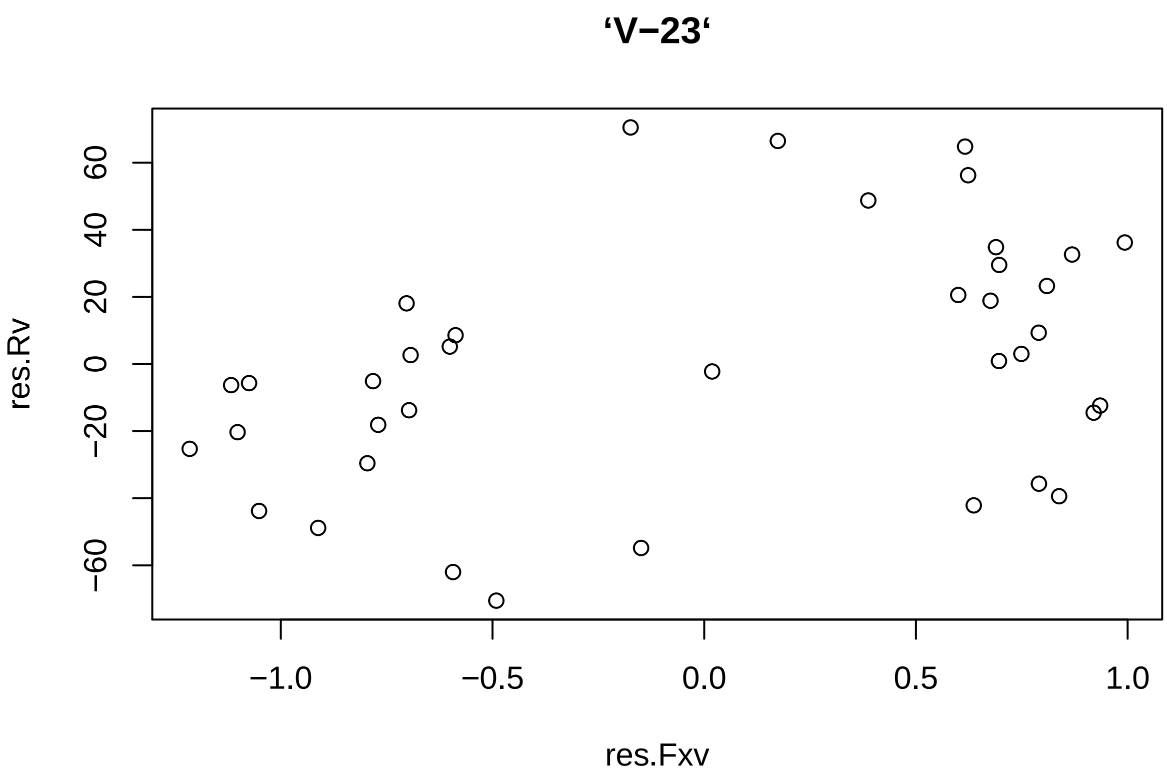
```
## 27 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 287.631579
## V-2            .
## V-3            .
## V-4            .
## V-5         144.960511
## V-6           6.192732
## V-7          23.099944
## V-8          29.472063
## V-11           .
## V-12           .
## V-13           .
## V-14           .
## V-15           .
## V-16           .
## V-17           .
## V-18           .
## V-19         14.891914
## V-20           .
## V-21           .
## V-22           .
## V-23         16.689107
## V-24           .
## V-25           .
## V-26           .
## V-27           .
## V-28           .
## V-29           .
```

```
# Model 3
# added variable factor to determine ^
datamodel3=data.frame(test.v10[,c(4,5,7,21,28)])
fitmodel3=lm(datamodel3$V.10~.,data = datamodel3)
colData3 <- list("`V-4`", "`V-5`", "`V-7`", "`V-23`")
names(colData3) <- c("`V-4`", "`V-5`", "`V-7`", "`V-23`")
removeXList <- colData3

for (rmX in removeXList){
  tmpV <- colData3
  tmpV[[rmX]] = NULL
  test.Rv=lm(as.formula(paste("`V-10` ~", paste(tmpV, collapse = "+"))), data = test.v10)
  res.Rv= test.Rv$residuals

  test.Fxv=lm(as.formula(paste(paste(rmX," ~"), paste(tmpV, collapse = "+"))), data = test.v10)
  res.Fxv= test.Fxv$residuals

  plot(res.Fxv,res.Rv,main = rmX)
}
```




'V−5'
200
100
0
−100
res.Rv
−0.5
0.0
0.5
1.0
res.Fxv


'V−7'
100
50
0
−50
res.Rv
−1
0
1
2
3
res.Fxv




```r
# Weighted tranformation for model 3
wts <- 1/fitted(lm(abs(residuals(fitmodel3)) ~ ., data = datamodel3))^2

fitmodel3weight <- lm(datamodel3$V.10~ .,data = datamodel3, weights=wts)
datamodel3weight=cbind(datamodel3[1:4],datamodel3$V.10*wts)
summary(fitmodel3weight)$r.squared
```

```
## [1] 0.9776405
```

```r
summary(fitmodel3weight)$adj.r.squared
```

```
## [1] 0.9749303
```

```r
#Brown test whether constant variance and transformation for Model 3 after tranformation
resmodel3b=fitmodel3weight $residuals
mmodel3=mean(datamodel3weight$`datamodel3$V.10 * wts`)
nmodel3=dim(datamodel3weight)[1]
#1. Break the residuals into two groups.
Group1 <- resmodel3b[datamodel3weight$`datamodel3$V.10 * wts`<mmodel3]
Group2 <-resmodel3b[datamodel3weight$`datamodel3$V.10 * wts`>=mmodel3]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group1)
M2 <- median(Group2)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group1 - M1 )) / length(Group1)
D2 <- sum( abs( Group2 - M2 )) / length(Group2)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group1 - M1) - D1 )^2 ) + sum( ( abs(Group2 - M2) - D2 )^2 ) ) / (nmodel3-2) )
```

```
#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group1) + 1/length(Group2) ) )
t
```

```
## [1] 1.422665
```

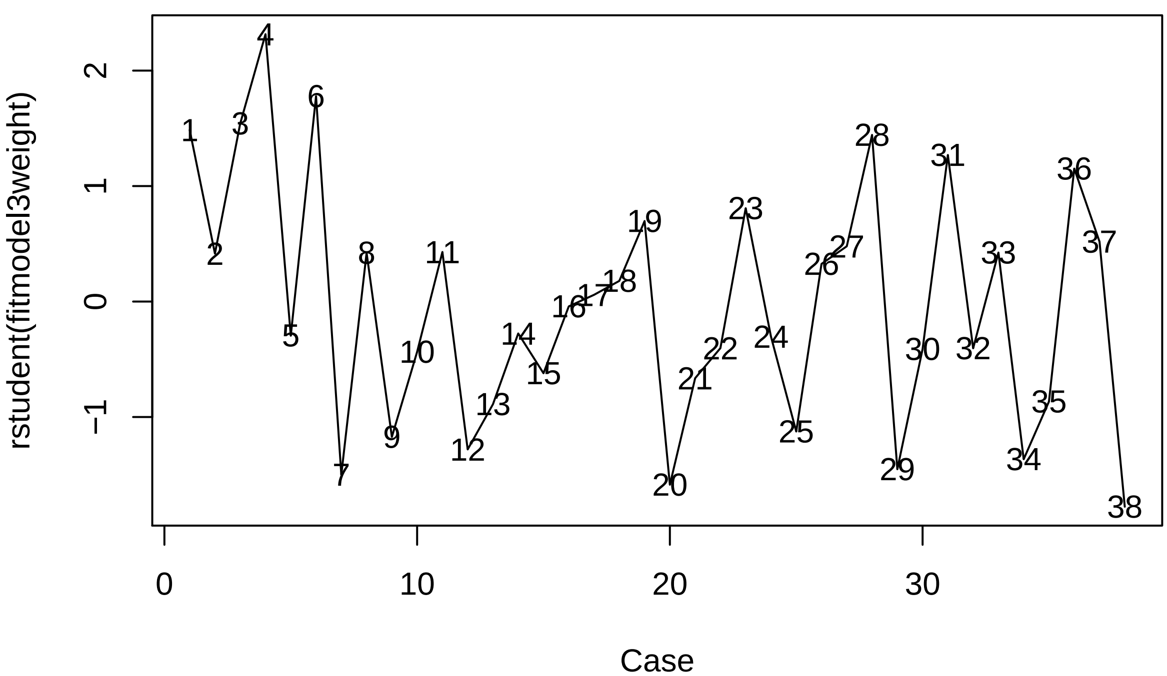
```
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel3-p1-1)   # find the catical value
```

```
## [1] 2.036933
```

```
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel3-p1-1))
```

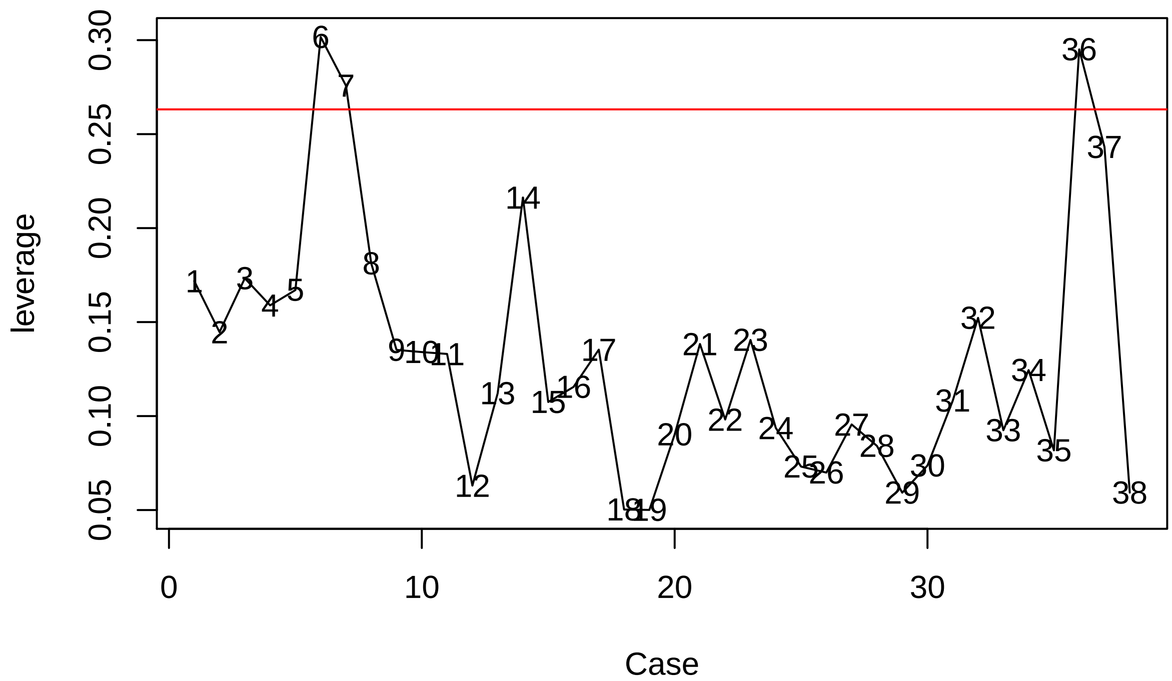
```
## [1] 0.1645093
```

```
#y outlier for model3
Case <- c(1:nmodel3)
plot(Case, rstudent(fitmodel3weight), type="l")
text(Case, rstudent(fitmodel3weight), Case)
```




```
alpha <- 0.05
crit <- qt(1-alpha/2/nmodel3, nmodel3-p1-1)
youtlier3=which(abs(rstudent(fitmodel3weight)) >=crit )
```

```
#x outlier for model3
X <- as.matrix(cbind(rep(1,nmodel3), datamodel3[1:4]))
H <- X%*%solve(t(X)%*%X, tol=1e-20)%*%t(X)
leverage <- hatvalues(fitmodel3weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p1/nmodel3, col=2)
```




```
xoutlier1=data.frame(which(leverage>2*p1/nmodel3) )
xoutlier1
```

```
##    which.leverage...2...p1.nmodel3.
## 6                                 6
## 7                                 7
## 36                               36
```

```
#test whether outlier in the extend of the model3
IM3=influence.measures(fitmodel3weight)
dxoutlier3=union(which(IM3$infmat[,8]>0.2),which(IM3$infmat[,6]>2*sqrt(p1/nmodel3)))
#combine x and y outlier
finaloutlier3=union(dxoutlier3,youtlier3)
datamodel3Final=datamodel3[-c(finaloutlier3),]
# get model1 without x y outlier
fitmodel3x1=lm(datamodel3Final$V.10~.,data = datamodel3Final)
wtsx3 <- 1/fitted(lm(abs(residuals(fitmodel3x1)) ~ ., data = datamodel3Final))^2
Fmodel3=lm(datamodel3Final$V.10~., data = datamodel3Final,weights =wtsx3)
# R2 & adj R2 for model3 test
summary(Fmodel3)$r.squared
```

```
## [1] 0.9797508
```

```
summary(Fmodel3)$adj.r.squared
```

```
## [1] 0.9770509
```

```
# add ^2 for model4
Data.new3 <- cbind(test.v10$`V-4`, test.v10$`V-5`, test.v10$`V-7`, test.v10$ `V-23`)
x3.new=as.matrix(cbind(Data.new3,((Data.new3)^2)[,-2]))
colnames(x3.new)=c("V-4","V-5","V-7","V-23","V-4.2","V-7.2","V-23.2")
```

```
# Model 4
testv14 = data.frame(x3.new,y)
datamodel4=data.frame(testv14[,c(2,8)])
```

```r
fitmodel4=lm(datamodel4$y~.,data = datamodel4)
summary(fitmodel4)$r.squared
```

```
## [1] 0.9500518
```

```r
summary(fitmodel4)$adj.r.squared
```

```
## [1] 0.9486644
```

```r
# Weighted tranformation for model 4
wts <- 1/fitted(lm(abs(residuals(fitmodel4)) ~ ., data = datamodel4))^2

fitmodel4weight <- lm(datamodel4$y~ .,data = datamodel4, weights=wts)
datamodel4weight=cbind(datamodel4[1],datamodel4$y*wts)
summary(fitmodel4weight)$r.squared
```

```
## [1] 0.9524123
```

```r
summary(fitmodel4)$adj.r.squared
```

```
## [1] 0.9486644
```

```r
#Brown test whether constant variance and transformation for Model 2 after tranformation
resmode22b=fitmodel4weight$residuals
mmodel4=mean(datamodel4weight$`datamodel4$y * wts`)
nmodel4=dim(datamodel4weight)[1]
#1. Break the residuals into two groups.
Group6 <- resmode22b[datamodel4weight$`datamodel4$y * wts`<mmodel4]
Group7 <- resmode22b[datamodel4weight$`datamodel4$y * wts`>=mmodel4]

#2. Obtain the median of each group, using the commands:
M1 <- median(Group6)
M2 <- median(Group7)

#3. Obtain the mean absolute deviation for each group, using the commands:
D1 <- sum( abs( Group6 - M1 )) / length(Group6)
D2 <- sum( abs( Group7 - M2 )) / length(Group7)

#4. Calculate the pooled standard error, using the command:
s <- sqrt( ( sum( ( abs(Group6 - M1) - D1 )^2 ) + sum( ( abs(Group7 - M2) - D2 )^2 ) ) / (nmodel4-2) )

#5. Finally, calculate the Brown-Forsythe test statistic, using the command:
t <- ( D1 - D2 ) / ( s * sqrt( 1/length(Group6) + 1/length(Group7) ) )
t
```

```
## [1] 1.582472
```

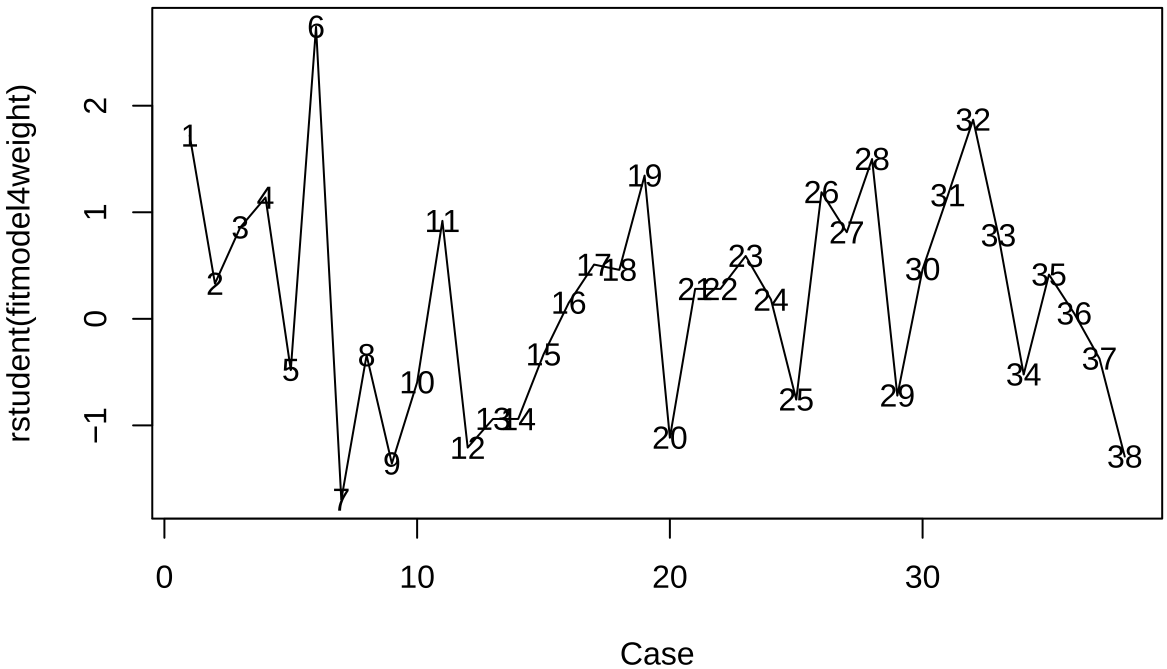
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel4-5)   # find the catical value
```

```
## [1] 2.034515
```

```r
# And the P-value can be found by typing:
 2*(1-pt( abs(t), nmodel4-5))
```

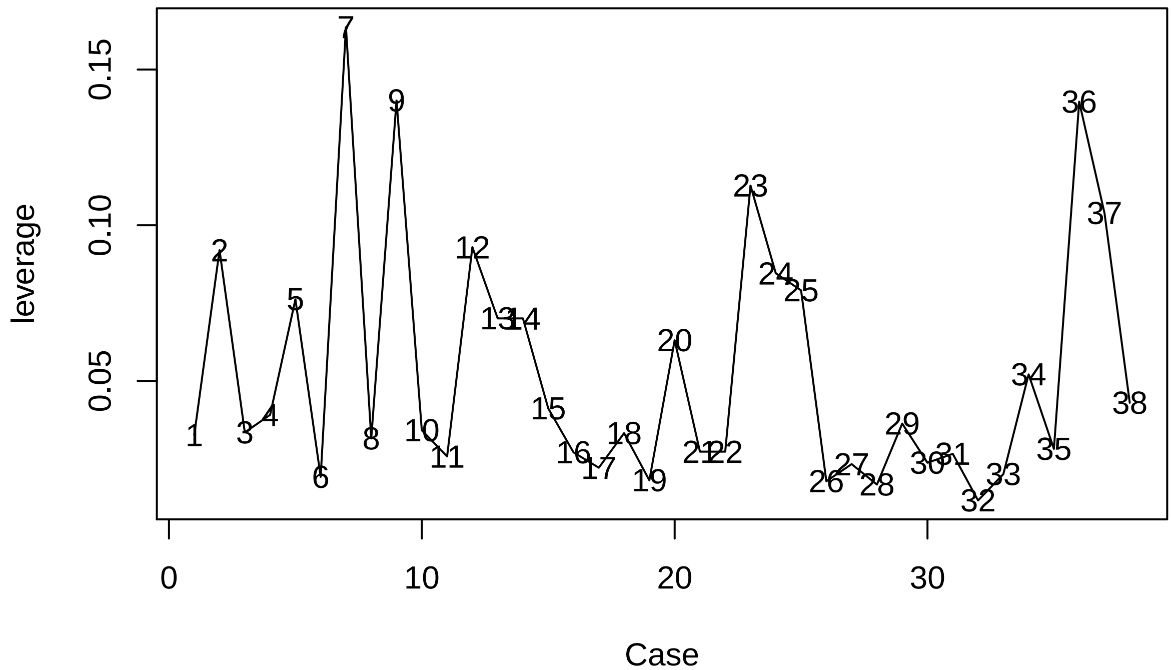
```
## [1] 0.1230801
```

```r
#y outlier
Case <- c(1:nmodel4)
plot(Case, rstudent(fitmodel4weight), type="l")
text(Case, rstudent(fitmodel4weight), Case)
```




```r
alpha <- 0.01
p=4
crit <- qt(1-alpha/2/nmodel4, nmodel4-p-1)
youtlier=which(abs(rstudent(fitmodel4weight)) >=crit )
```

```r
#x outlier
X <- as.matrix(cbind(rep(1,nmodel4), datamodel4weight[1]))
H <- X%*%solve(t(X)%*%X,tol=1e-30)%*%t(X)
leverage <- hatvalues(fitmodel4weight)
plot(Case, leverage, type="l")
text(Case, leverage, Case)
abline(h=2*p/nmodel4, col=2)
```




```
xoutlier=data.frame(which(leverage>2*p/nmodel4) )
xoutlier
```

```
## [1] which.leverage...2...p.nmodel4.
## <0 rows> (or 0-length row.names)
```

```
#test whether outlier in the extend of the model
IM4=influence.measures(fitmodel4weight)
dxoutlier=union(which(IM4$infmat[,5]>0.2),which(IM4$infmat[,3]>2*sqrt(p/nmodel4)))
#combine x and y outlier
finaloutlier=union(dxoutlier,youtlier)
datamodel4Final=datamodel4[-c(finaloutlier),]
# get model2 without x y outlier
fitmodel4x2=lm(datamodel4Final$y~.,data = datamodel4Final)
wtsx2 <- 1/fitted(lm(abs(residuals(fitmodel4x2)) ~ ., data = datamodel4Final))^2
Fmodel4=lm(datamodel4Final$y~., data = datamodel4Final,weights =wtsx2)
# R2 & adj R2 for model1
summary(Fmodel4)$r.squared
```

```
## [1] 0.9516841
```

```
summary(Fmodel4)$adj.r.squared
```

```
## [1] 0.9503037
```