STAT 350 Final Project of predicting V-9

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
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
```

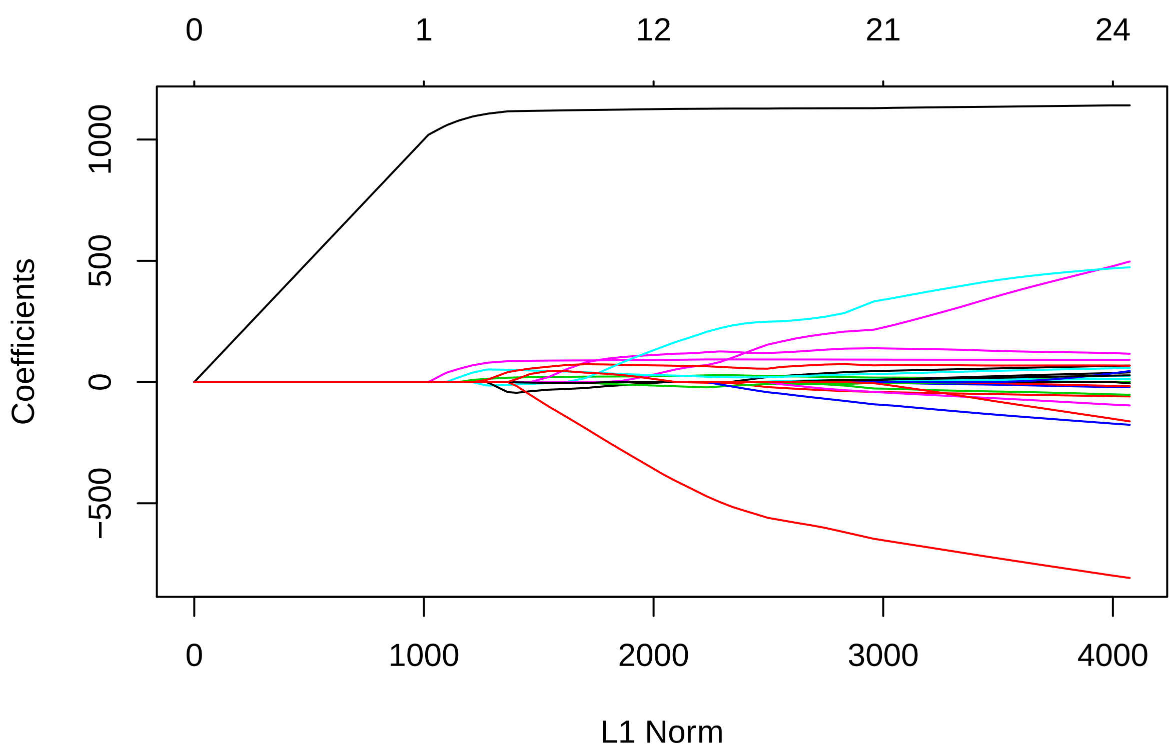
```
## V-25  0.781298186 -0.248323465  0.977863958  0.97321585  0.88823633
## V-26  0.792637630 -0.283334553  0.981947612  0.97801582  0.89700001
## V-27  0.570548168 -0.069144581  0.824139284  0.81998426  0.91703848
## V-28  0.561385894  0.037568332  0.296720859  0.31415987  0.25026572
## V-29  0.828660567 -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
```

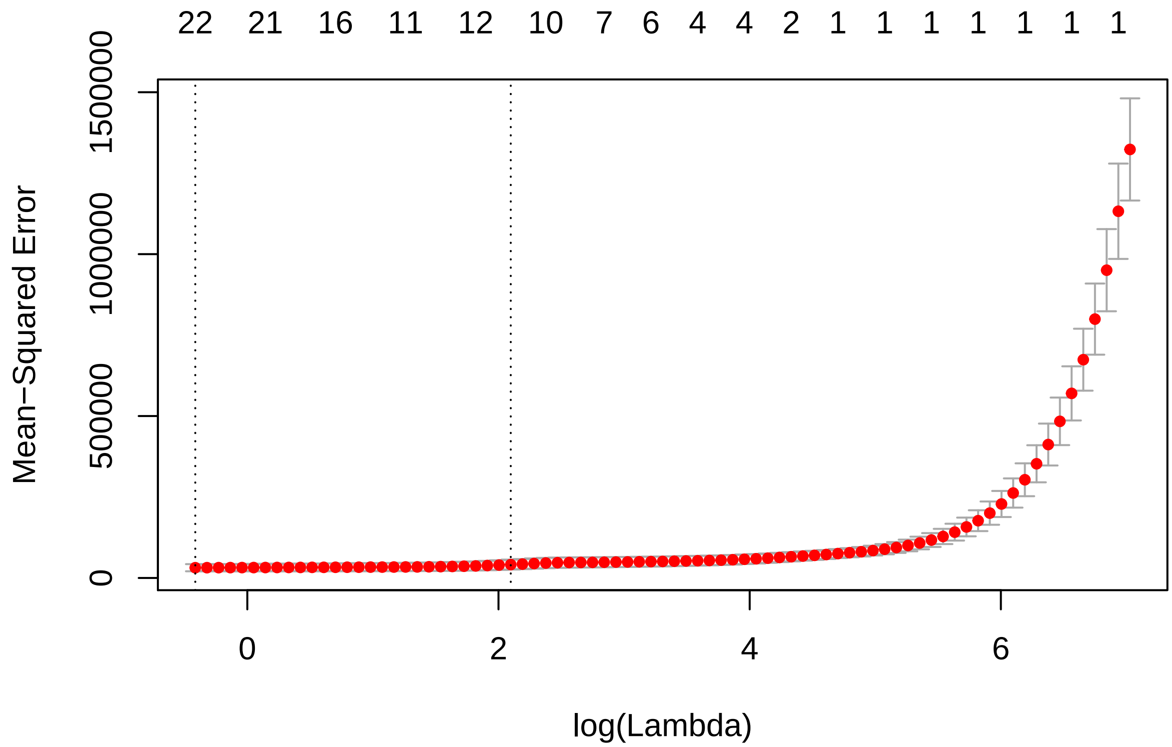
```
# Lasso to determine variable
library("glmnet")
```

```
## 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
```

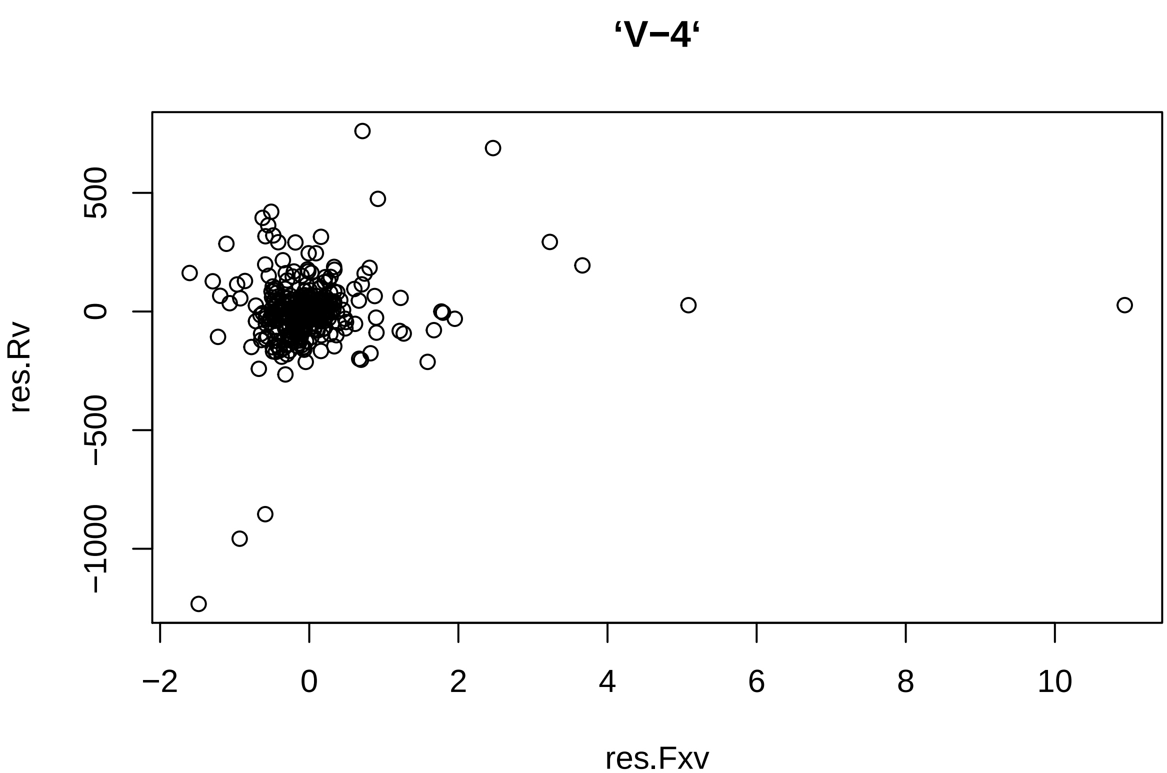
```r
# added variable factor to determine ^
datamodel1=data.frame(train.v9[,c(4,7,8,14,15,16,18,19,21,24,26,27,28)])
# Model 1
fitmodel1=lm(datamodel1$V.9~.,data = datamodel1)
summary(fitmodel1)$r.squared
```

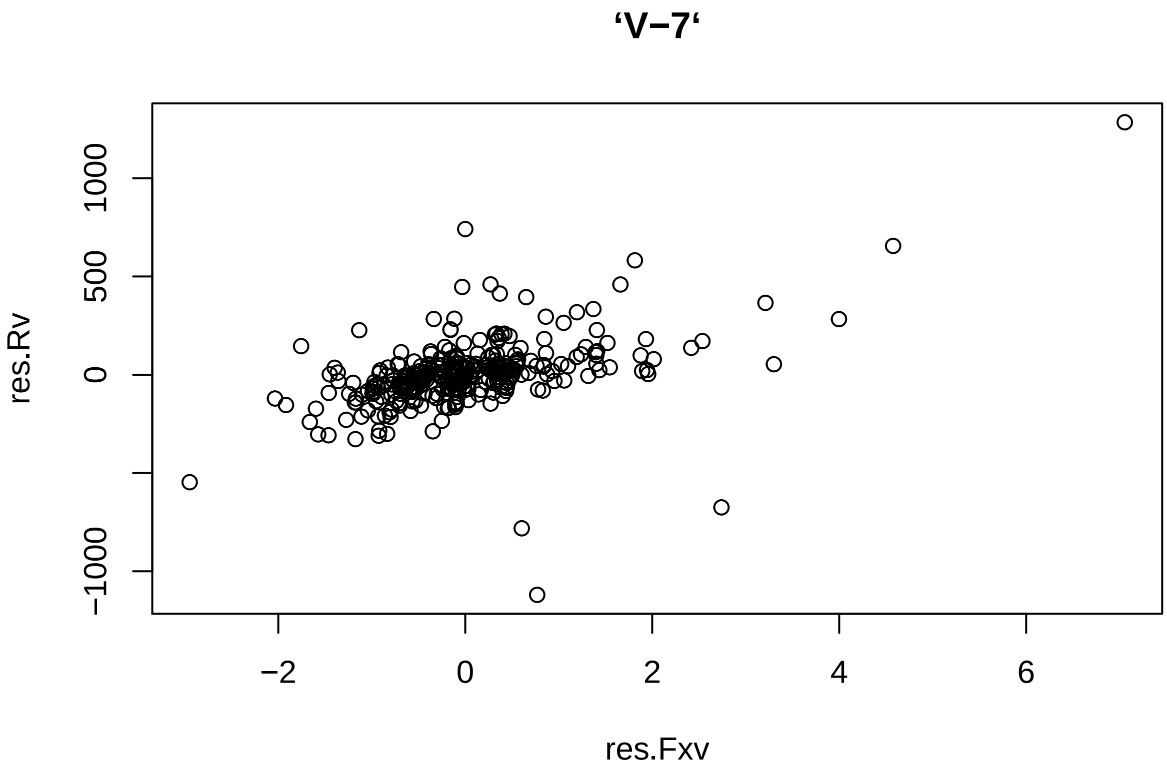
```
## [1] 0.9825749
```

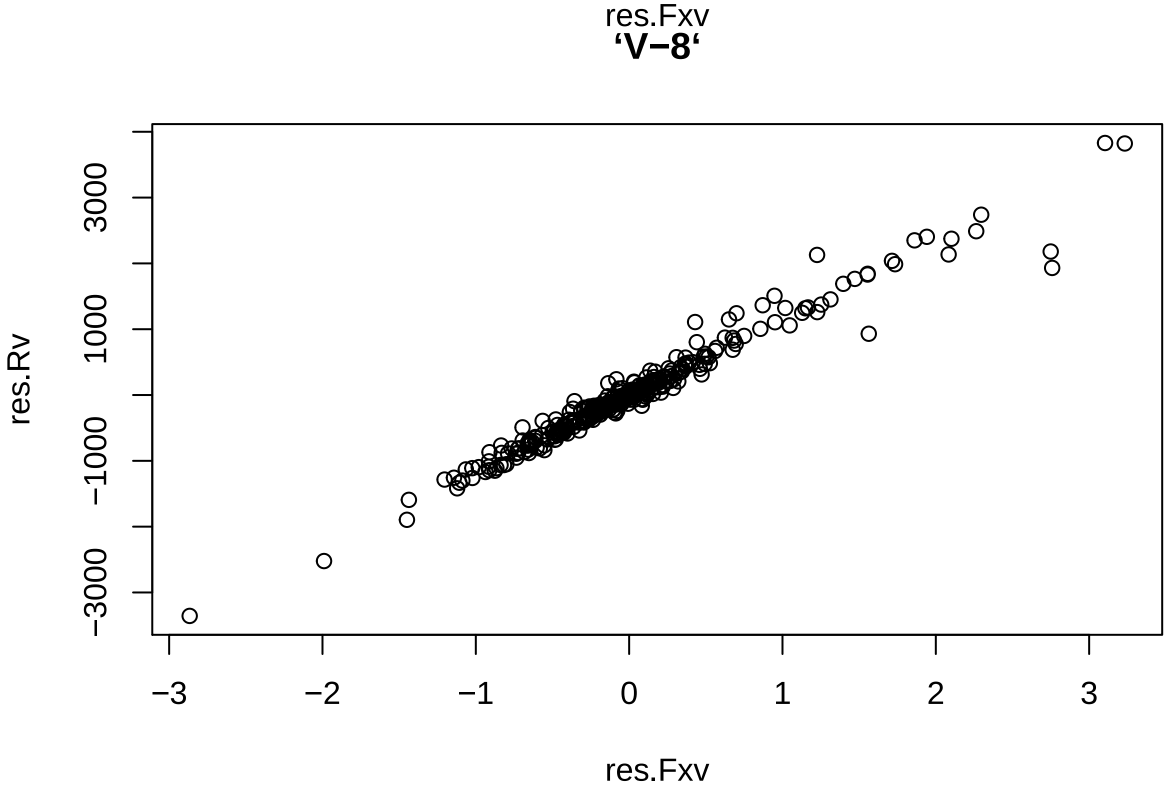
```r
summary(fitmodel1)$adj.r.squared
```

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

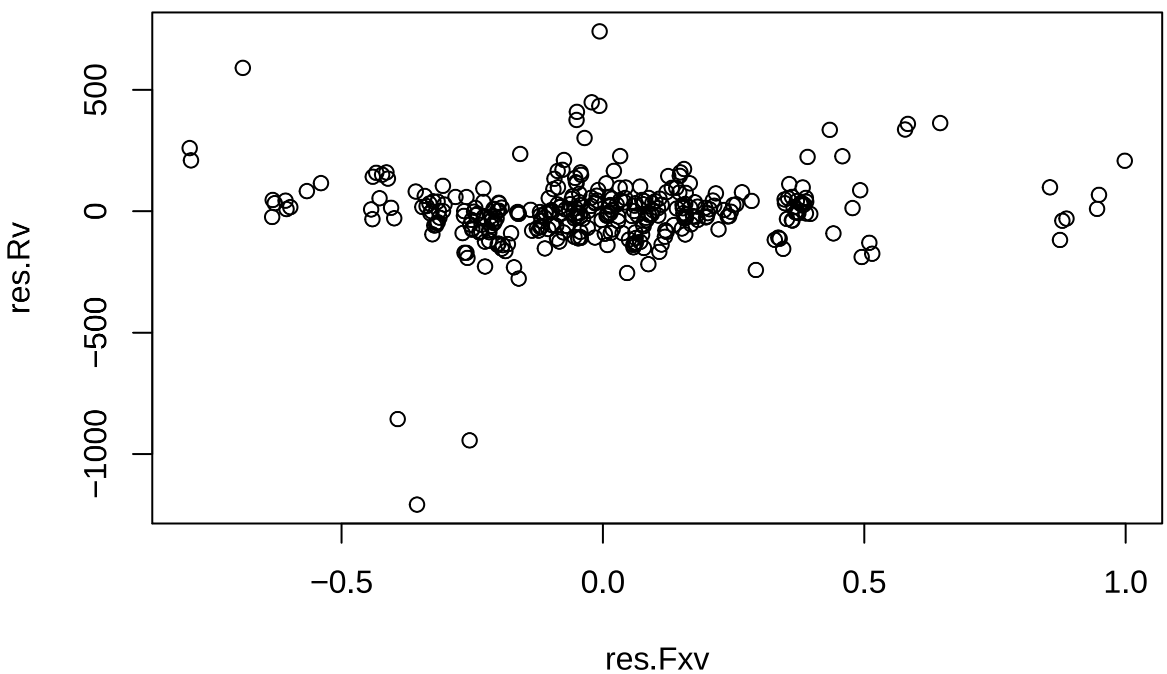
```r
colData <- list("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
names(colData) <- c("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
removeXList <- colData

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
res.Fxv
1000
500
0
−1000
−2
0
2
4
6

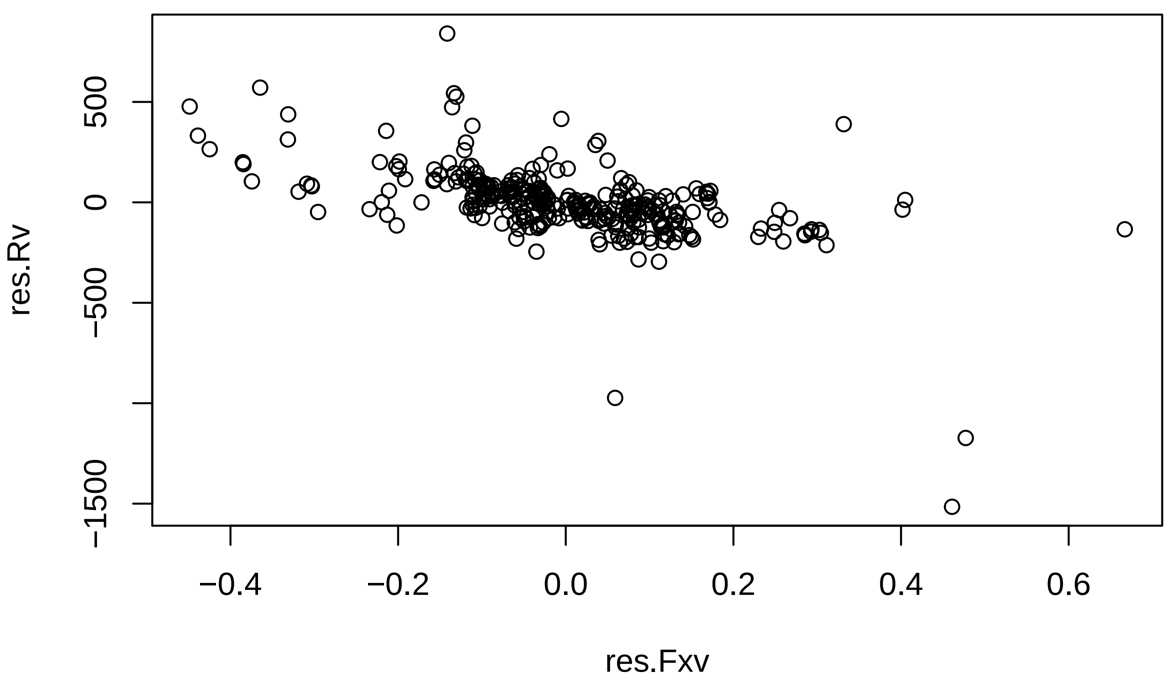

‘V−8‘
res.Rv
res.Fxv
3000
1000
−1000
−3000
−3
−2
−1
0
1
2
3

‘V−16‘

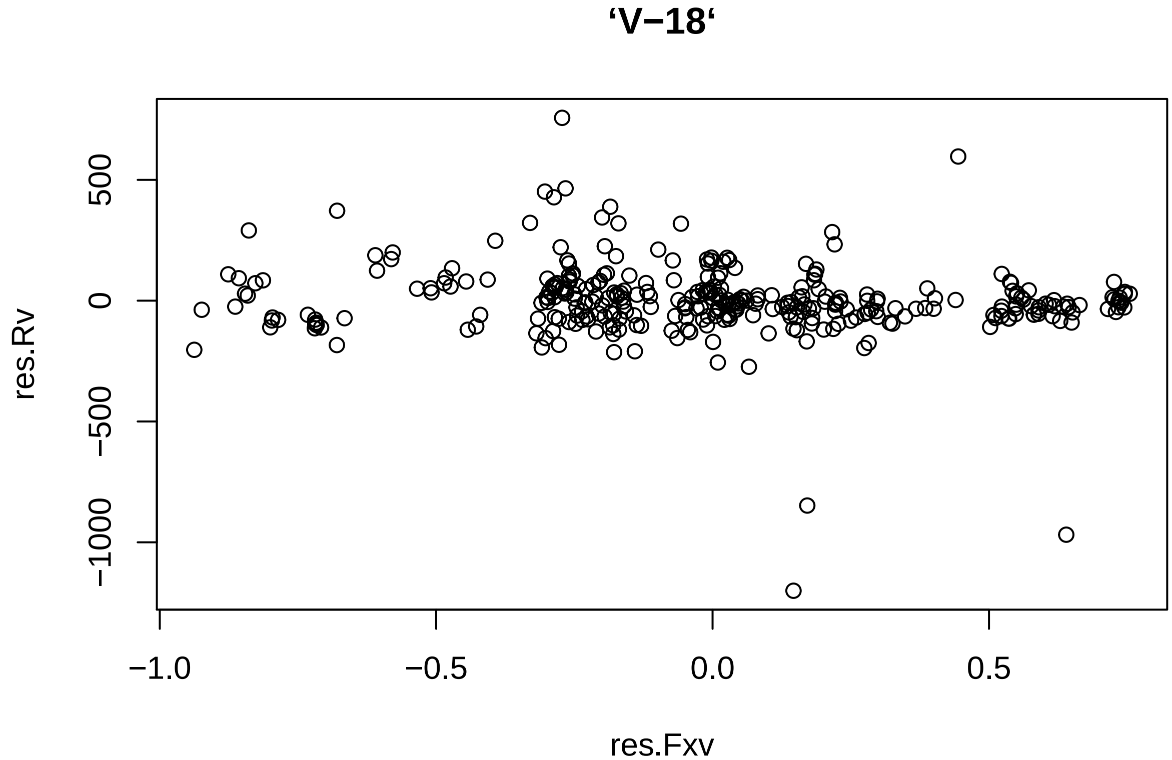


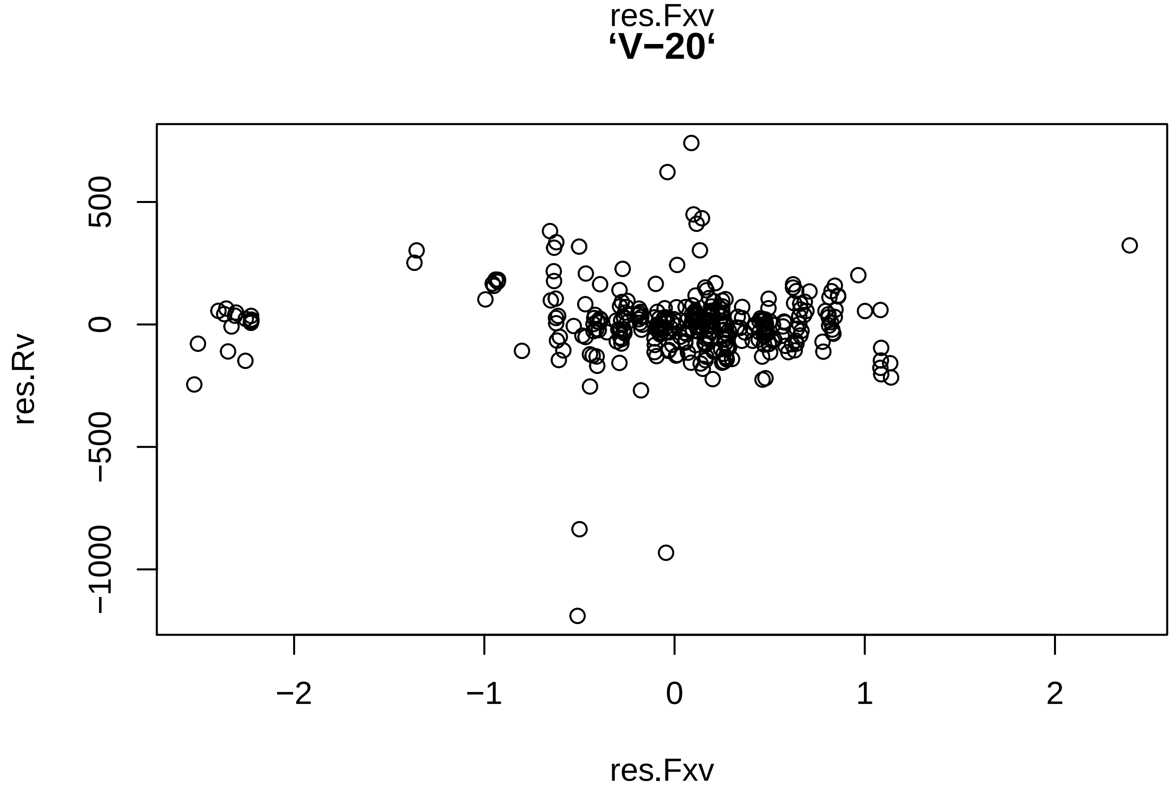
500
0
−500
−1000
res.Rv
−0.5
0.0
0.5
1.0
res.Fxv

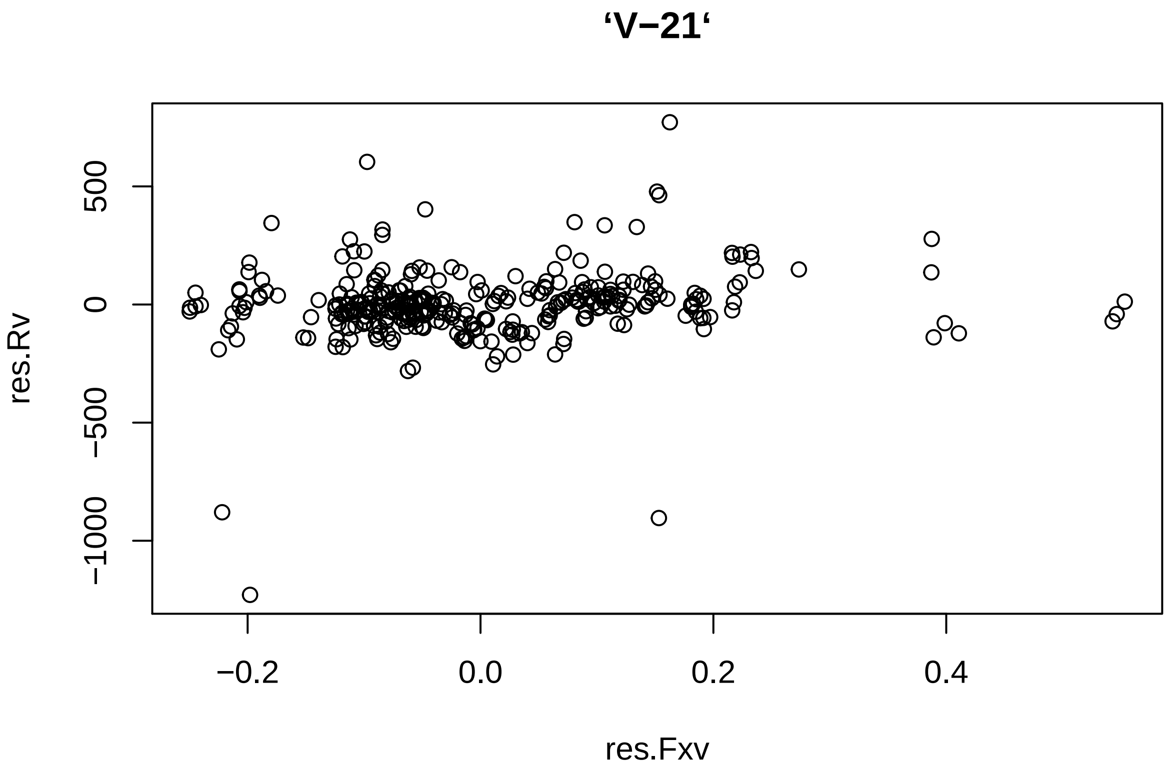
‘V−17‘

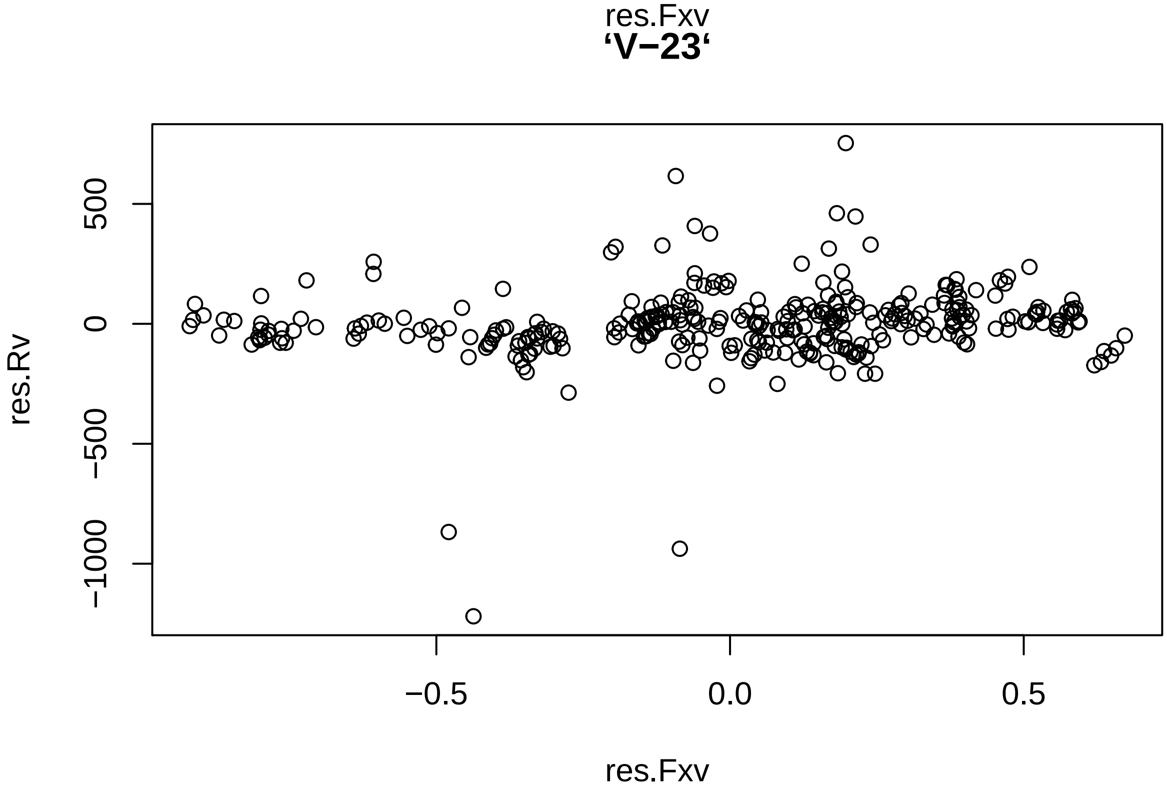


500
0
−500
−1500
res.Rv
−0.4
−0.2
0.0
0.2
0.4
0.6
res.Fxv

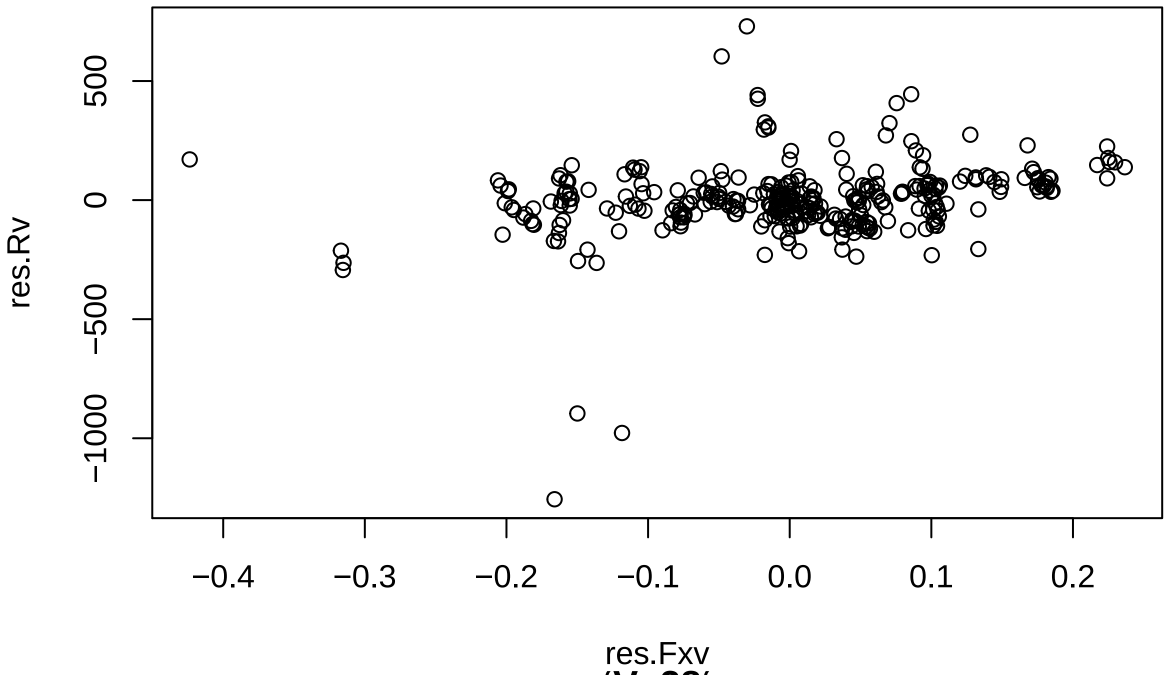

‘V−18‘
res.Rv
500
0
−500
−1000
−1.0
−0.5
0.0
0.5
res.Fxv


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

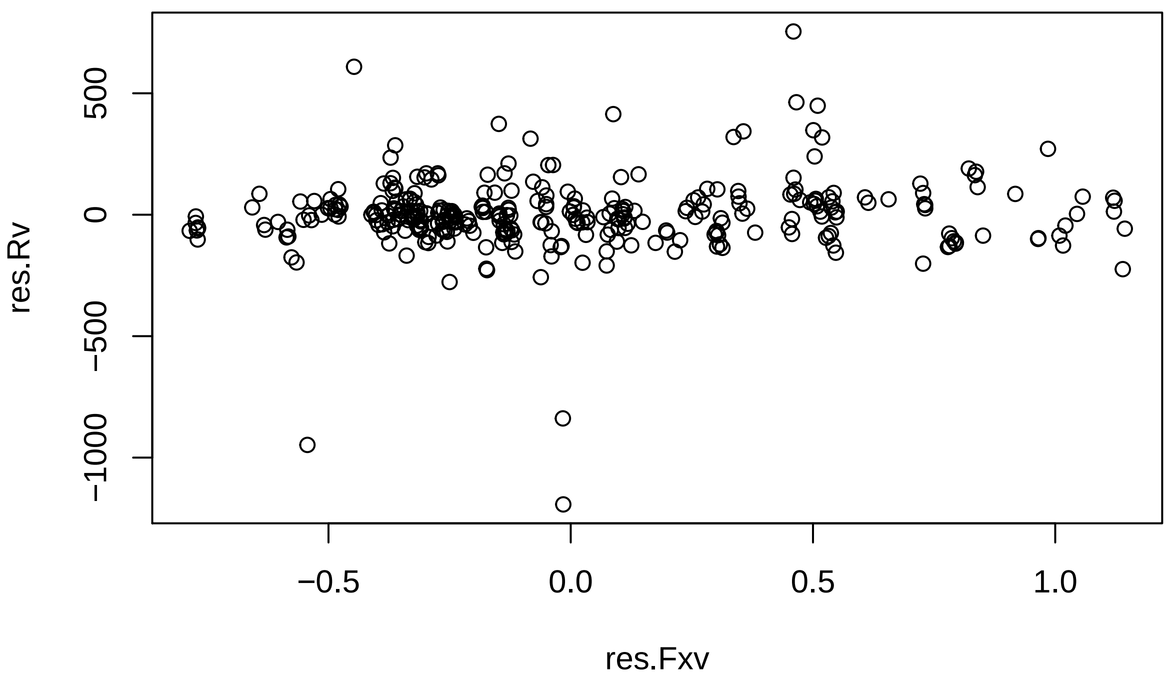

‘V−21‘
500
0
−500
−1000
res.Rv
−0.2
0.0
0.2
0.4
res.Fxv

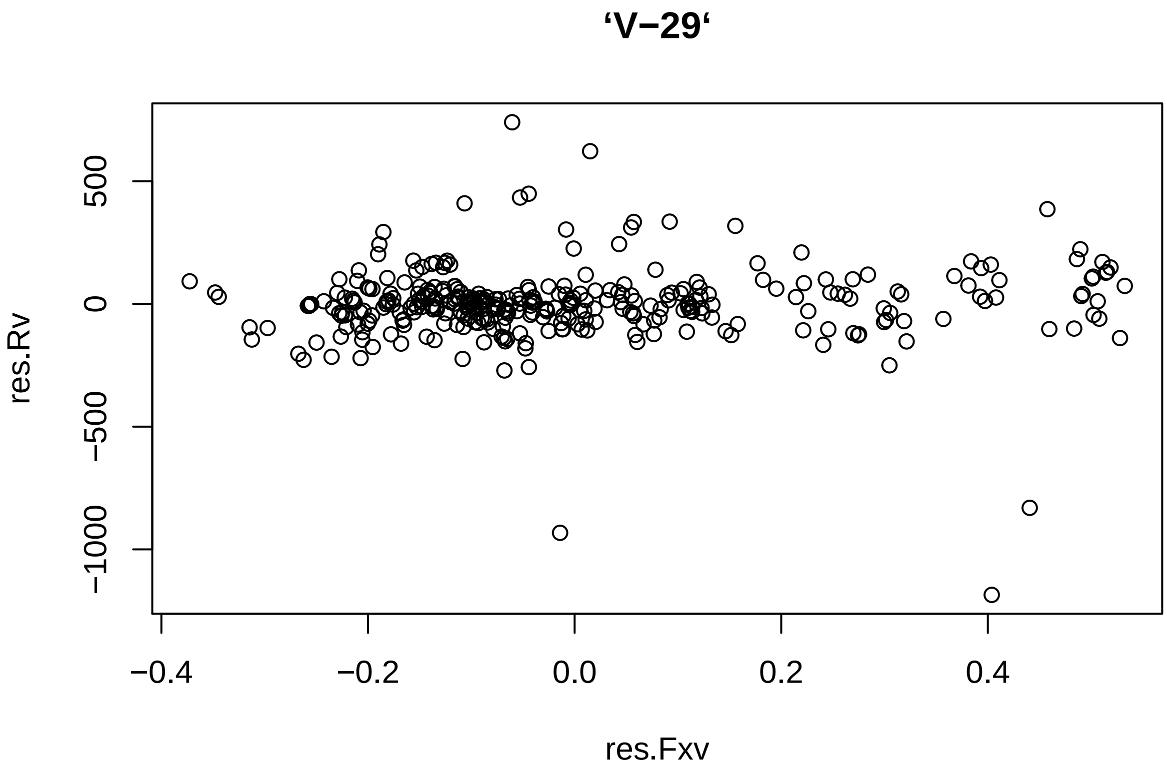

‘V−23‘
500
0
−500
−1000
res.Rv
−0.5
0.0
0.5
res.Fxv

‘V−26‘


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

‘V−28‘


500
0
−500
−1000
res.Rv
−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
```

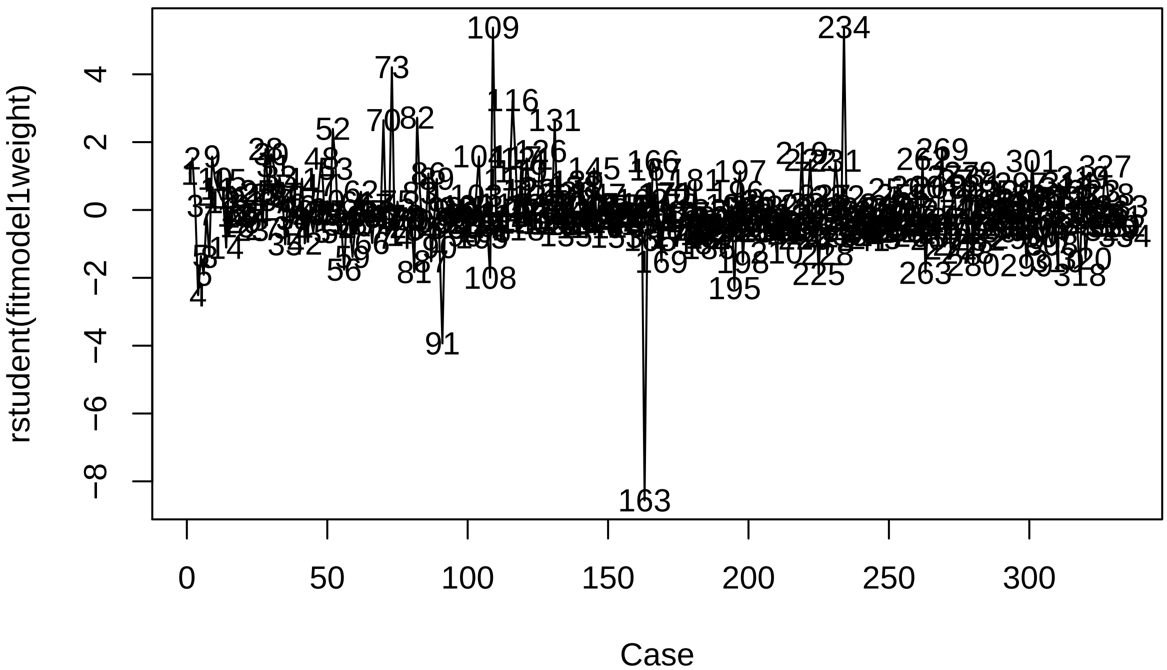
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)   # find the catical value
```

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

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

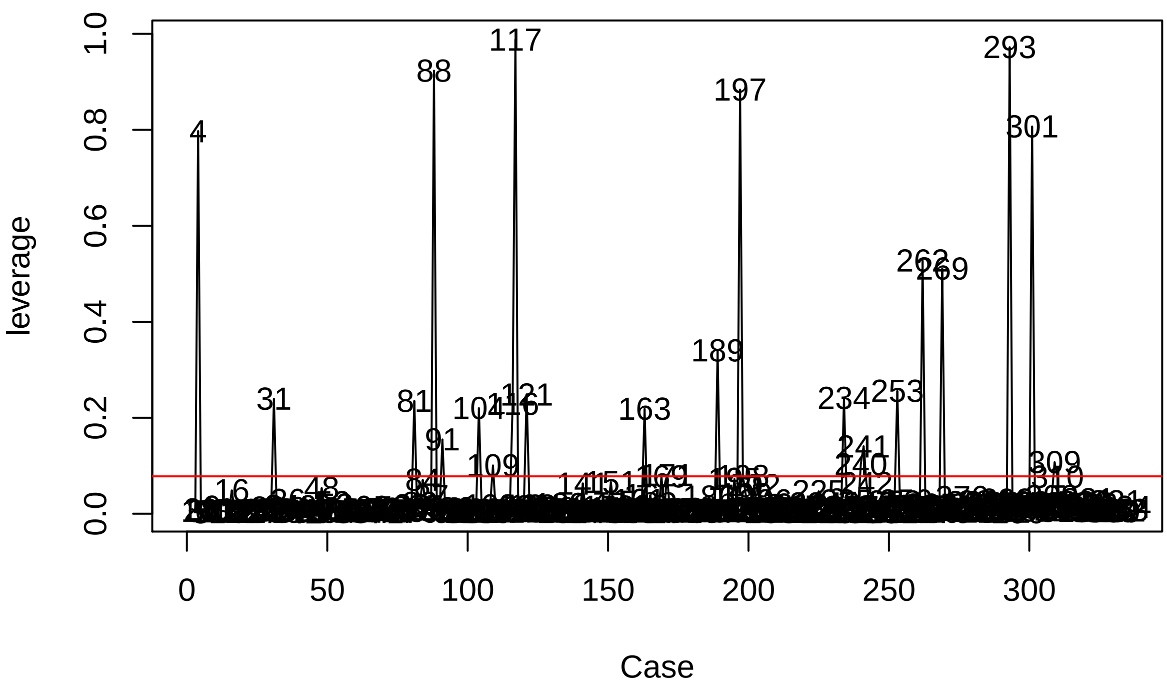
```
## [1] 0.1524126
```

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




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

```r
#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.
## 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
```

```
#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
```

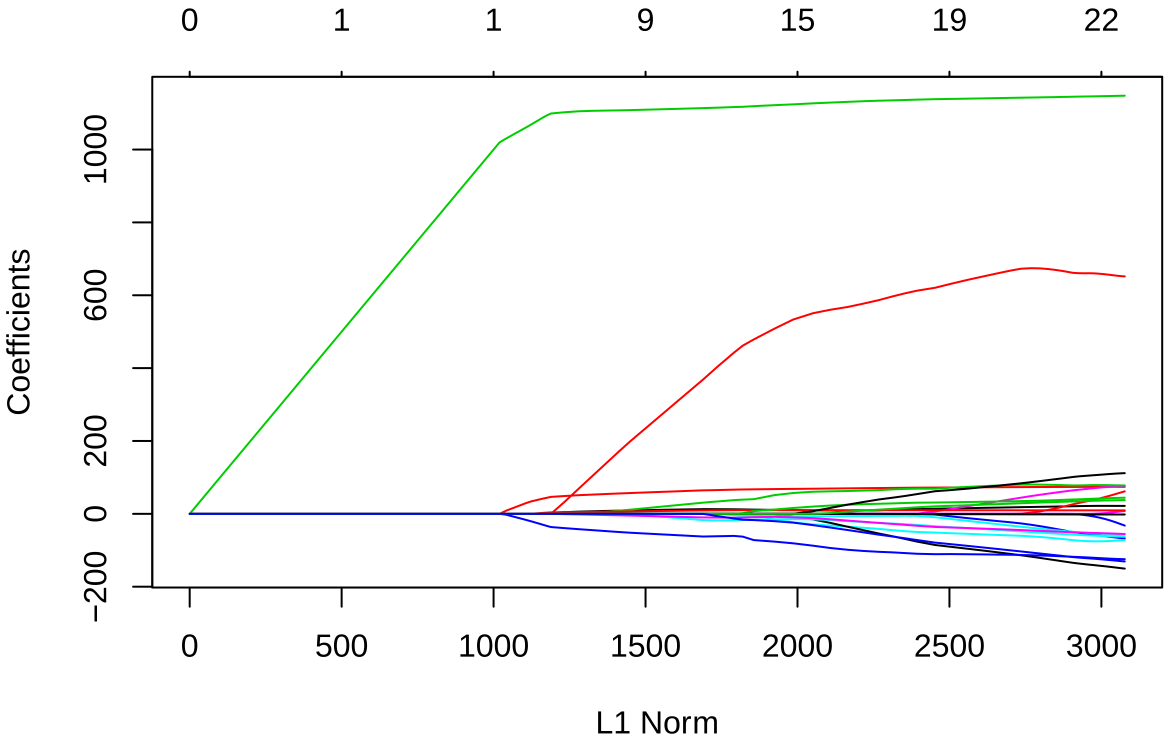
```
summary(Fmodel1)$adj.r.squared
```

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

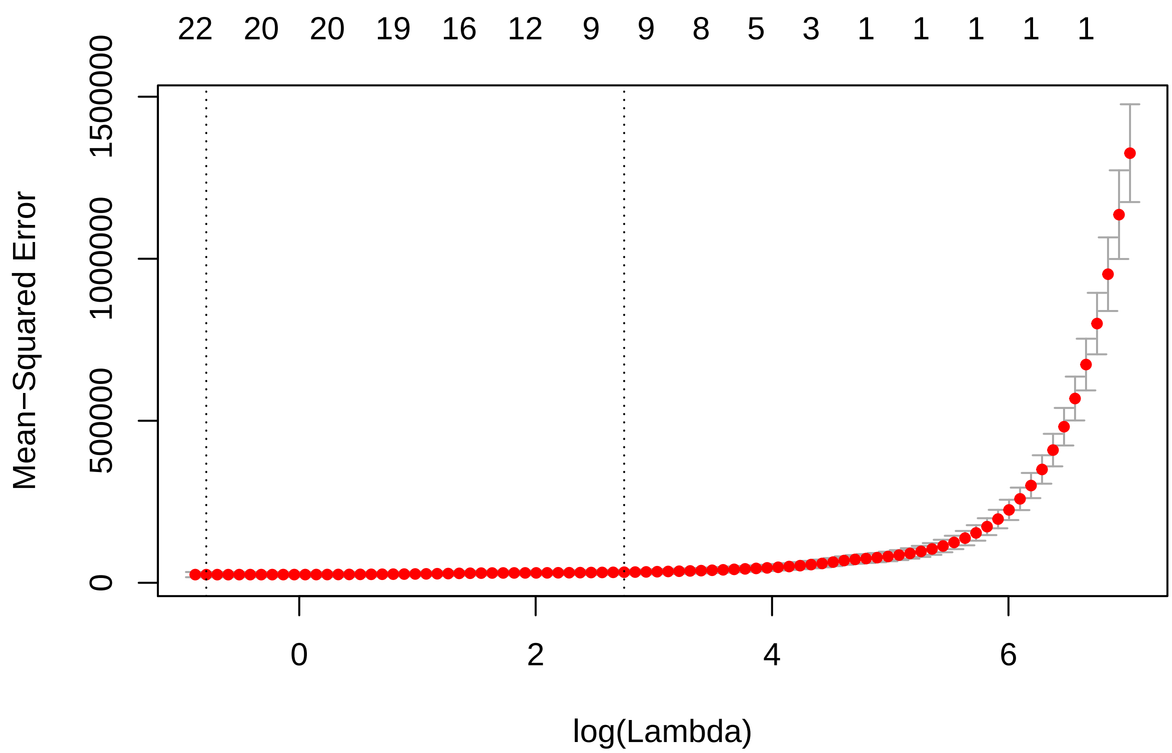
```
# add ^2 for model2
Data.new <- cbind(train.v9$`V-4`, train.v9$`V-7`, train.v9$`V-8`, train.v9$ `V-16`, train.v9$`V-17`, tra
x2.new=as.matrix(cbind(Data.new,((Data.new)^2)[,-3]))
colnames(x2.new)=c("V-4","V-7","V-8","V-16","V-17","V-18","V-20","V-21","V-23","V-26","V-28","V-29","V-4
```

```
#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         .
```

```r
# 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
```

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

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

```r
#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)
```

```
#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
```

```
#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
```

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

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

```
# 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
```

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

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

```
#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
```

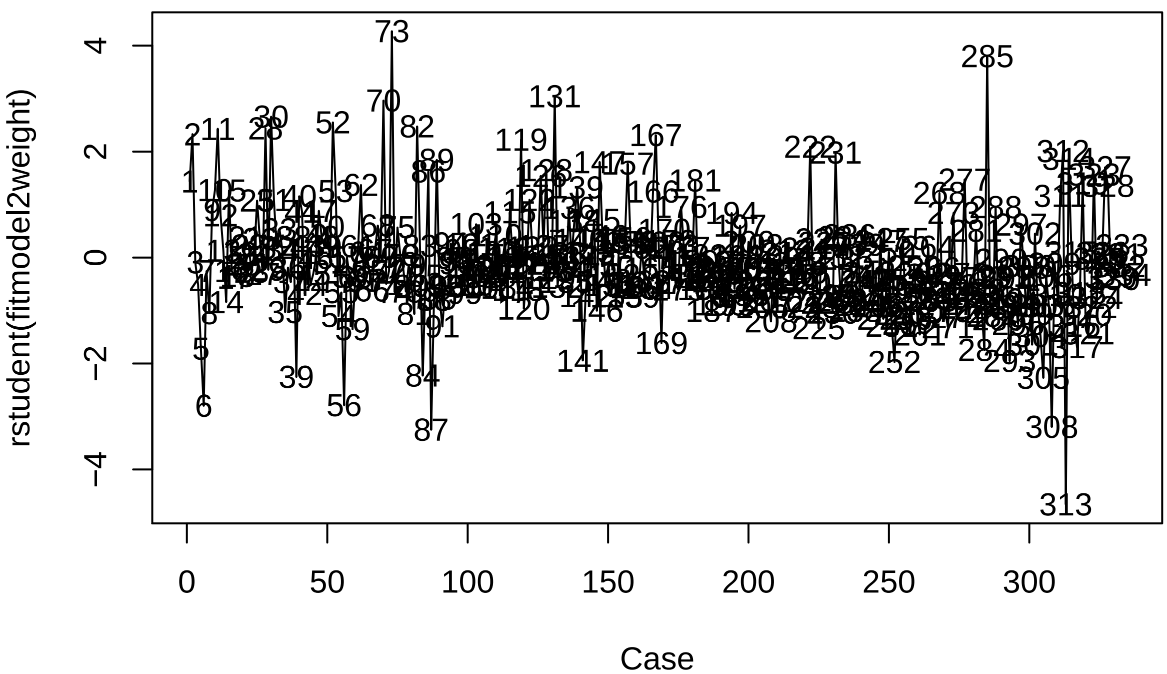
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel2-17)   # find the catical value
```

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

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

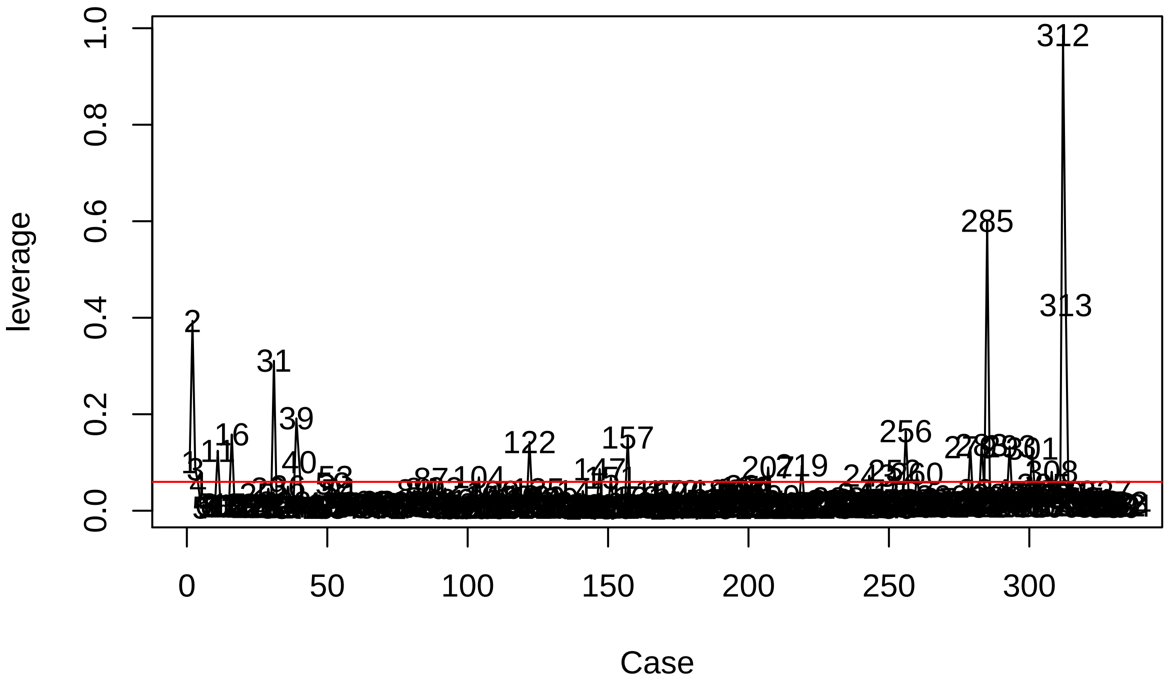
```
## [1] 0.4689819
```

```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.
## 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
```

```r
#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])
}
```

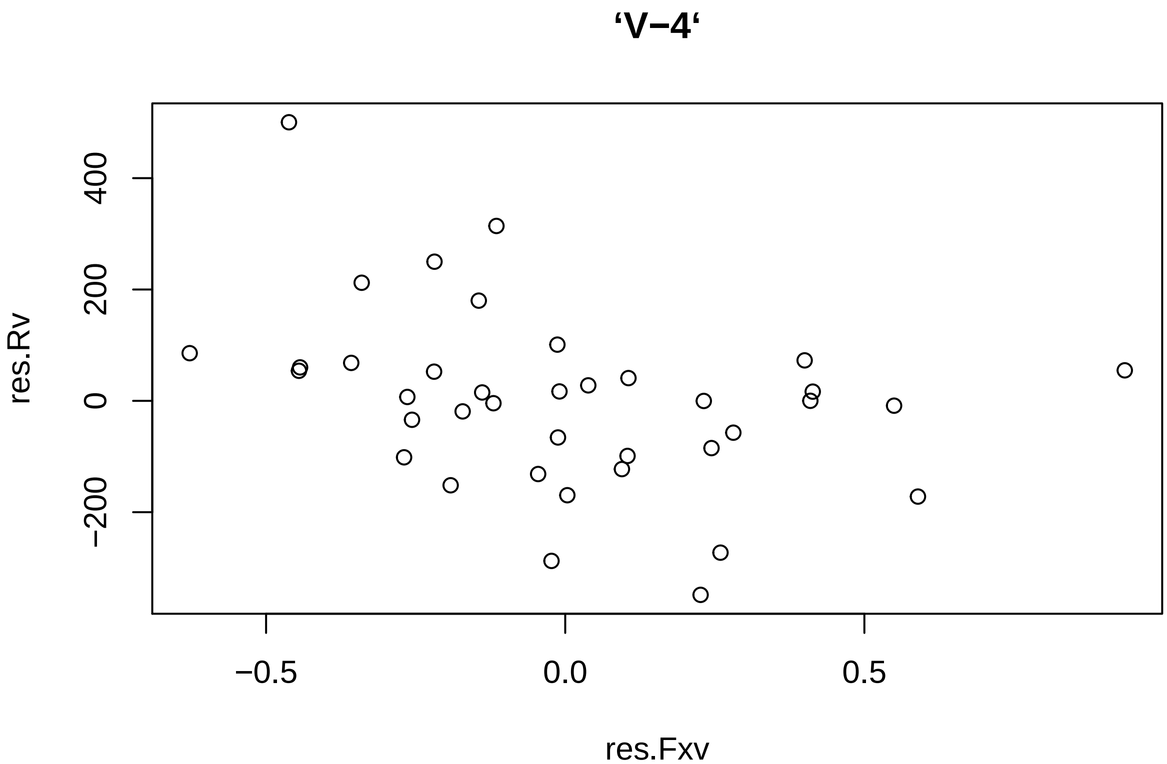
```r
# added variable factor to determine ^
datamodel1=data.frame(test.v9[,c(4,7,8,14,15,16,18,19,21,24,26,27,28)])
# Model 1
fitmodel1=lm(datamodel1$V.9~.,data = datamodel1)
summary(fitmodel1)$r.squared
```

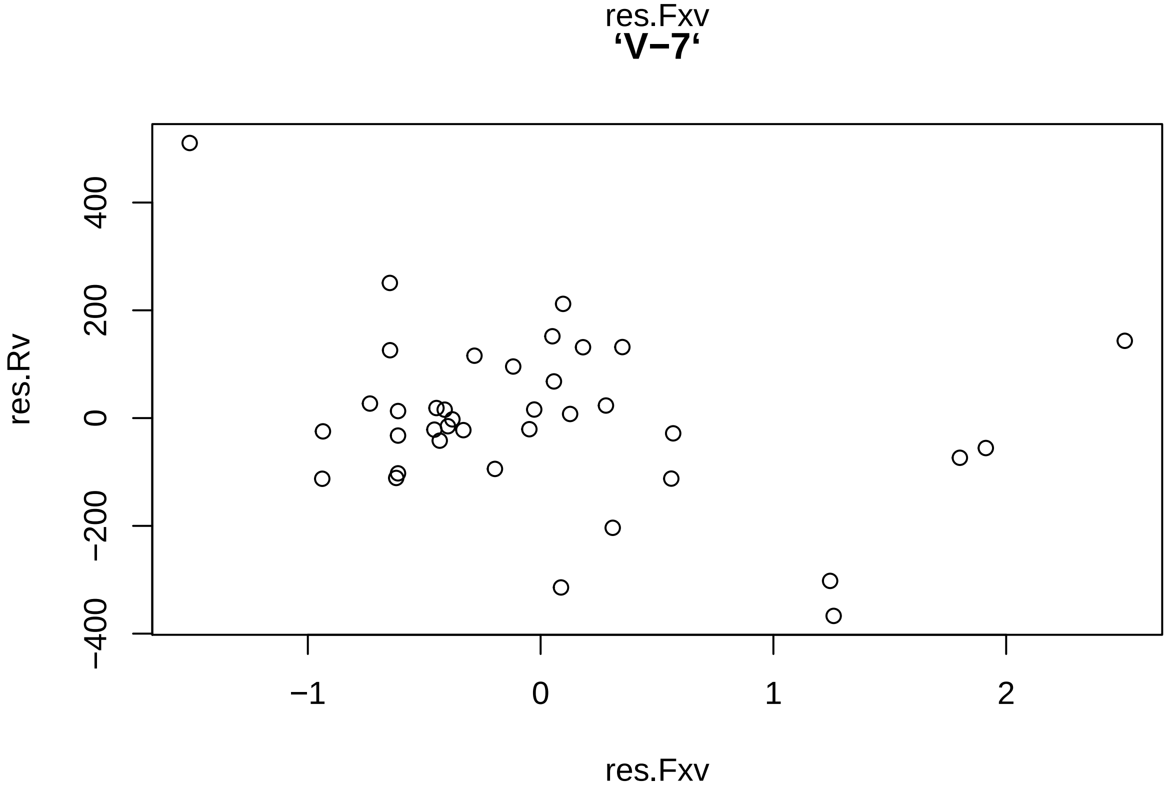
```
## [1] 0.9909658
```

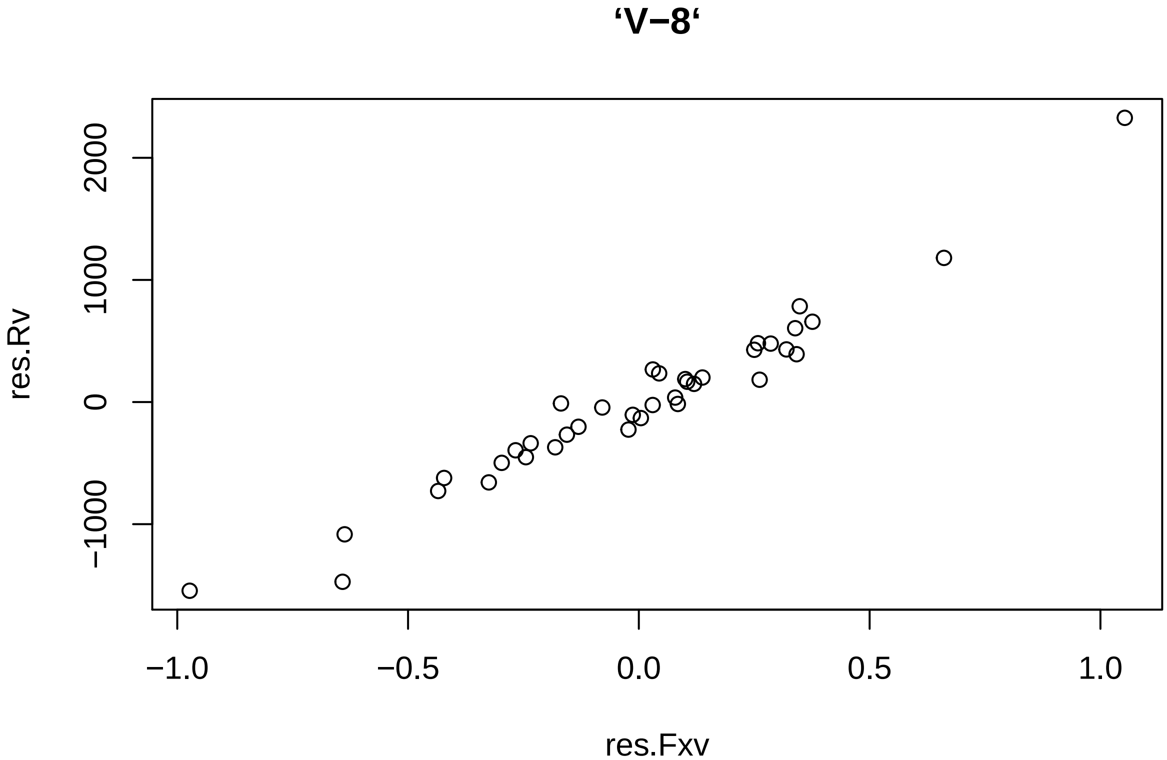
```r
summary(fitmodel1)$adj.r.squared
```

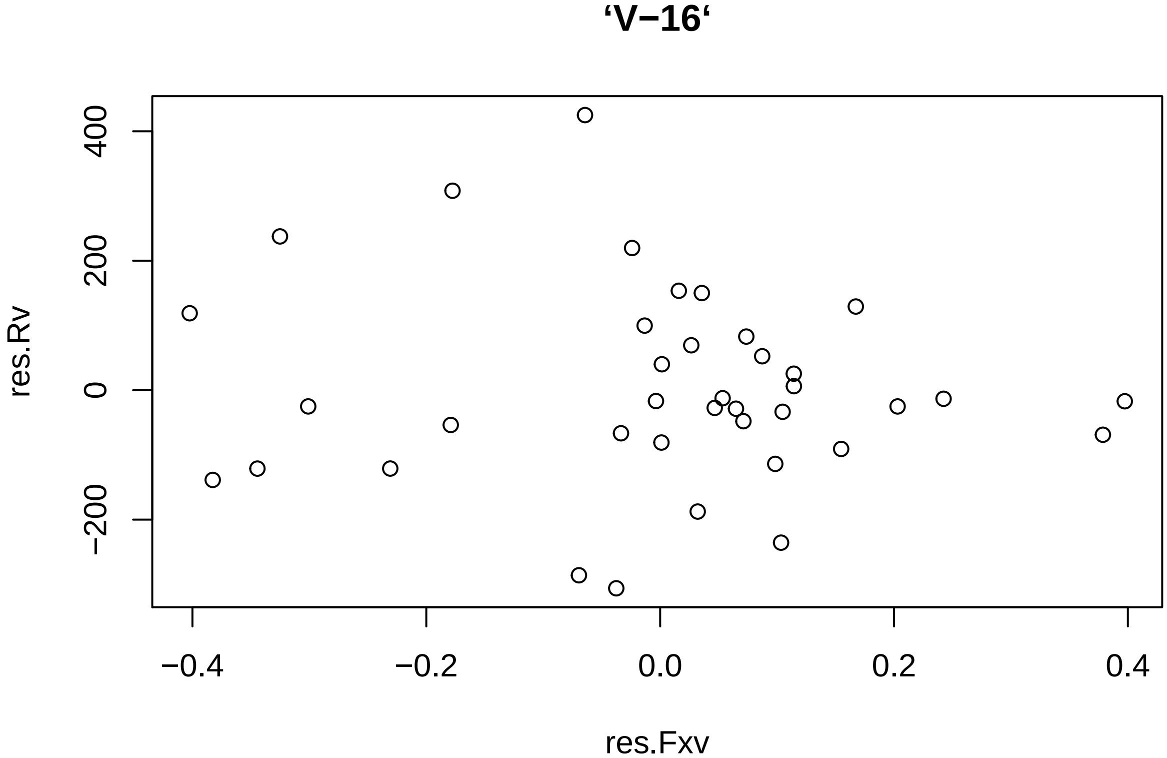
```
## [1] 0.9866294
```

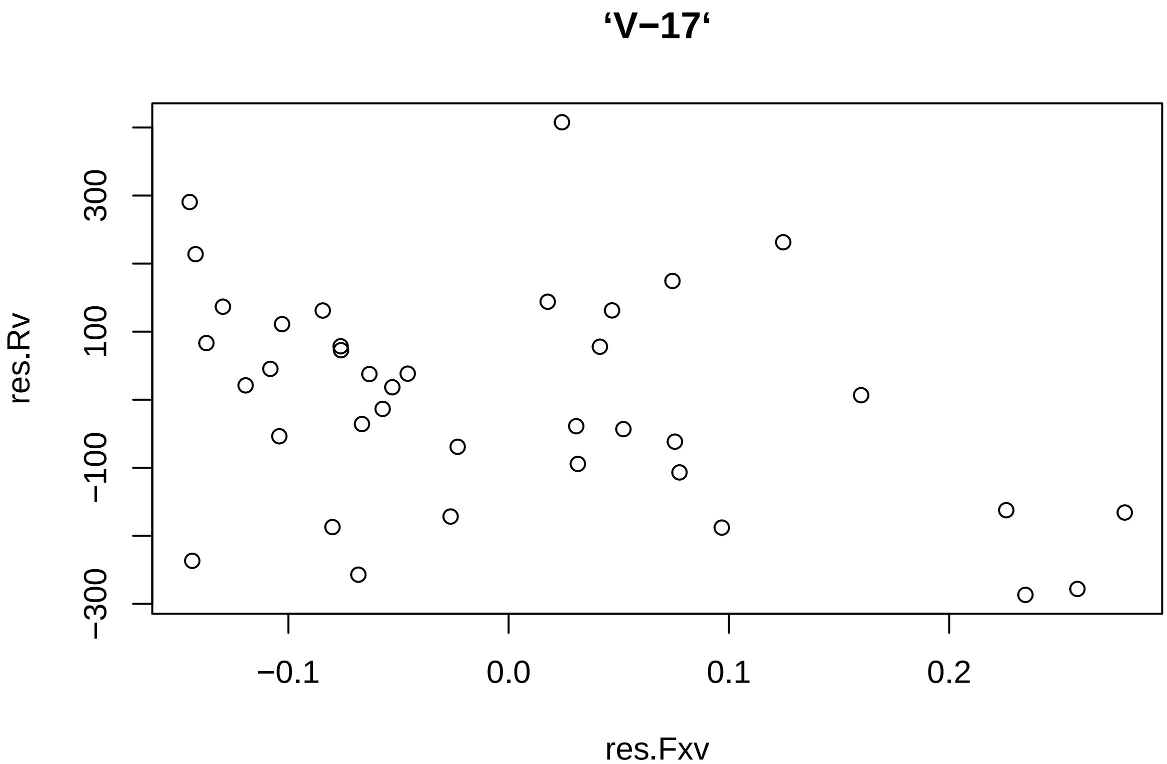
```r
colData <- list("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
names(colData) <- c("`V-4`", "`V-7`", "`V-8`", "`V-16`",
    "`V-17`", "`V-18`", "`V-20`", "`V-21`", "`V-23`", "`V-26`", "`V-28`", "`V-29`")
removeXList <- colData

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

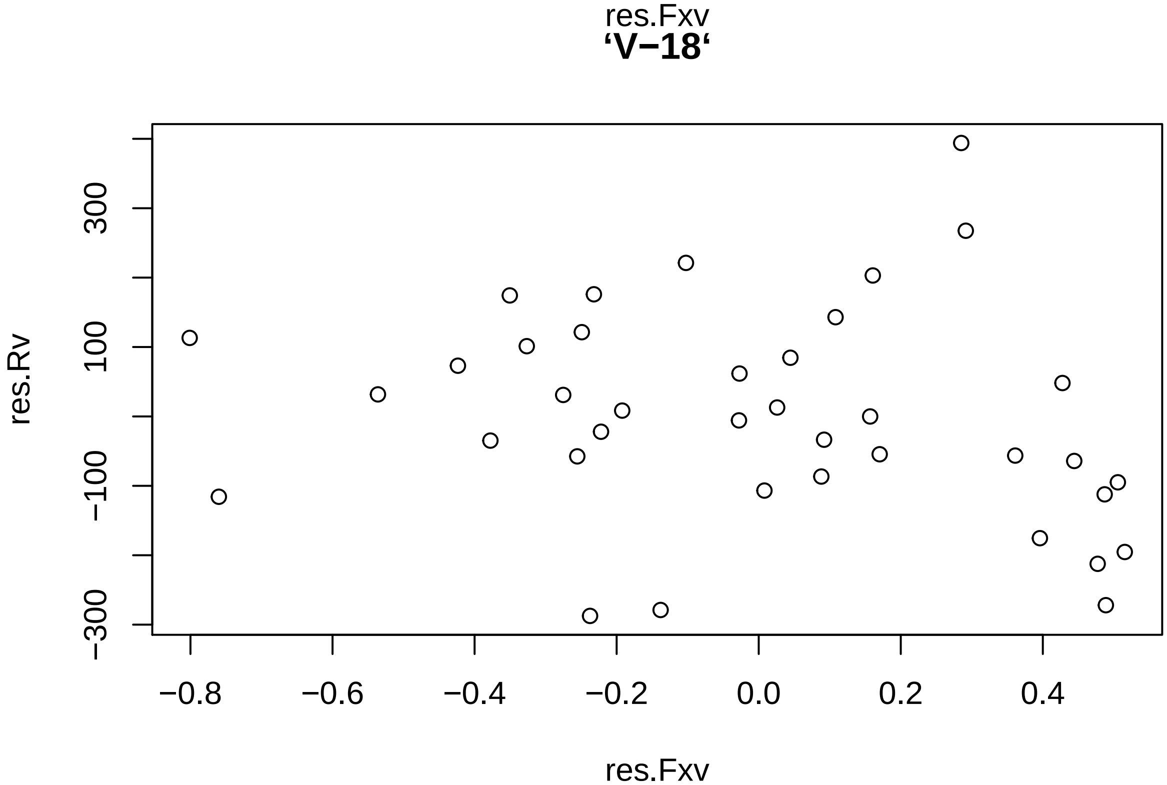

‘V−4‘
res.Rv
400
200
0
−200
−0.5
0.0
0.5
res.Fxv

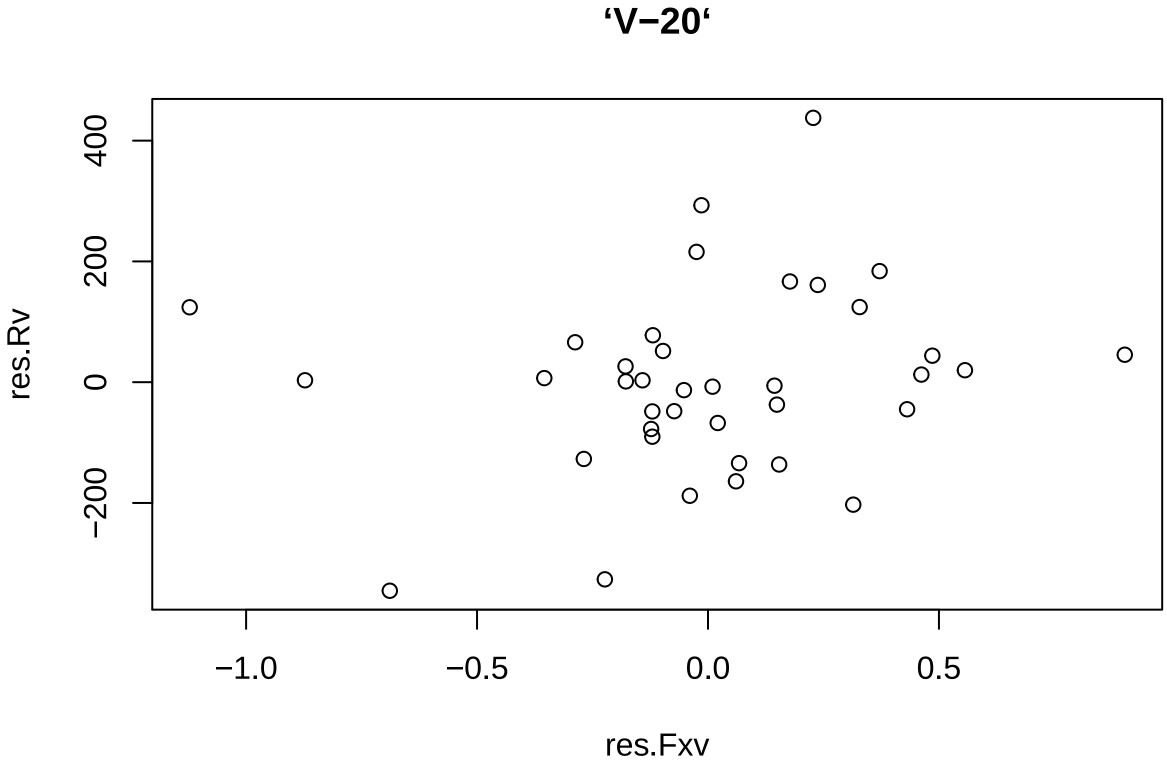

‘V−7‘
res.Rv
400
200
0
−200
−400
−1
0
1
2
res.Fxv

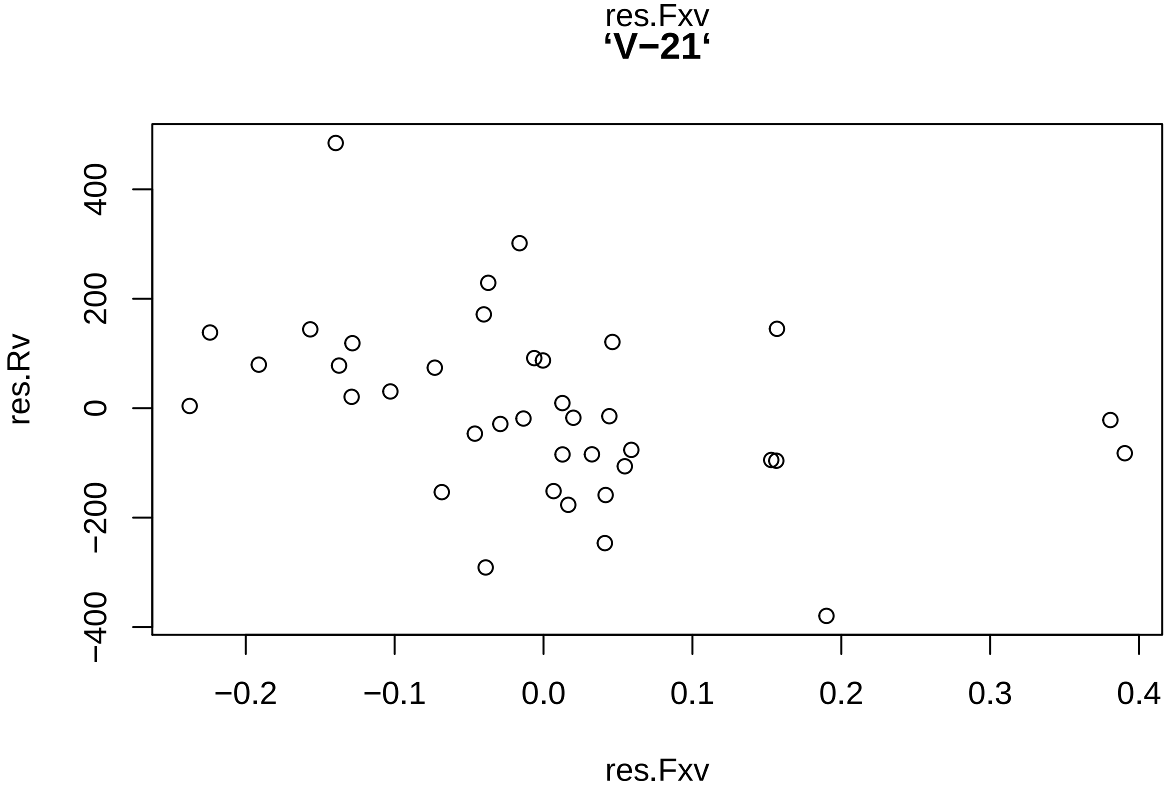

'V−8'
res.Rv
res.Fxv
−1000
0
1000
2000
−1.0
−0.5
0.0
0.5
1.0

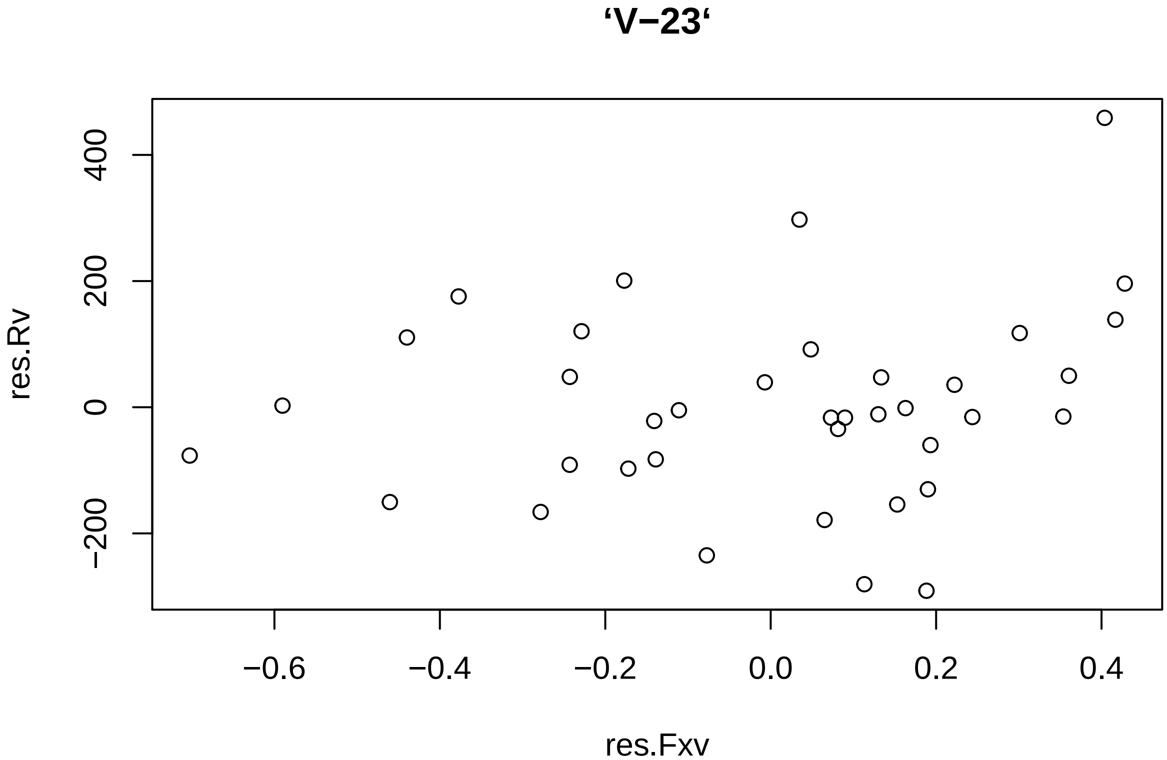

'V−16'
res.Rv
res.Fxv
−200
0
200
400
−0.4
−0.2
0.0
0.2
0.4

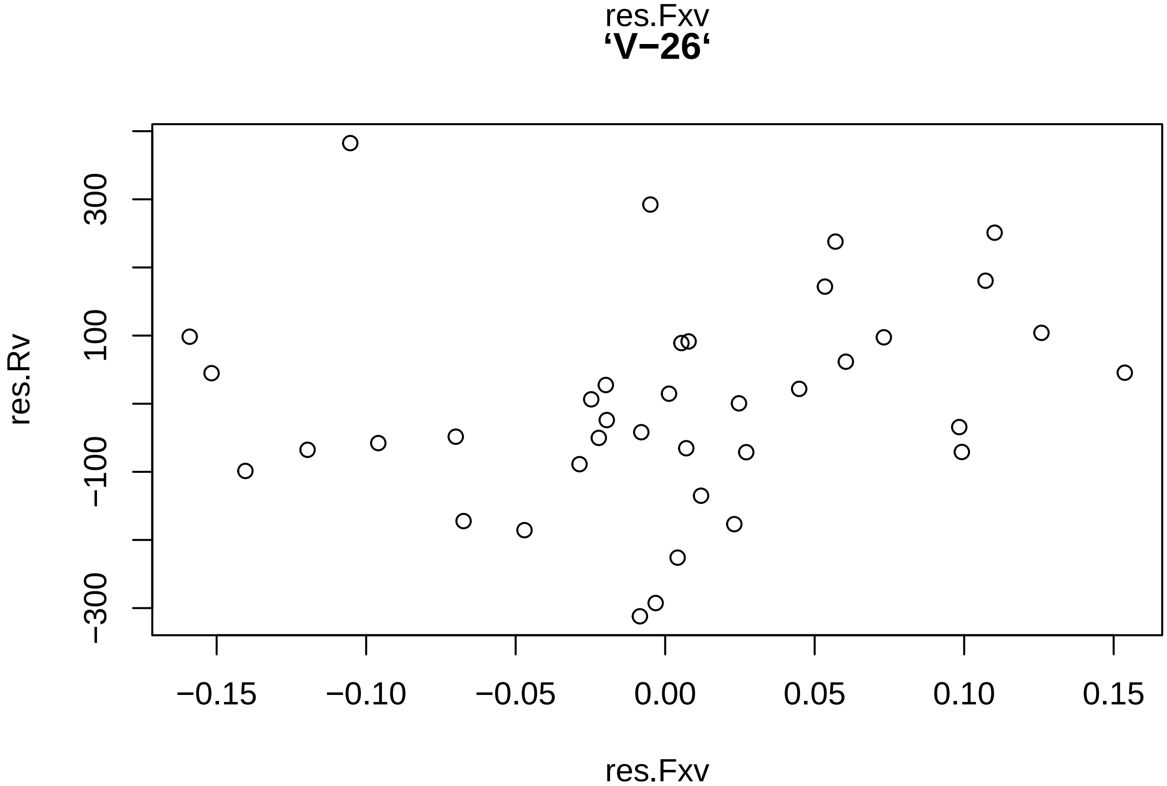

‘V−17‘
res.Rv
300
100
−100
−300
−0.1
0.0
0.1
0.2
res.Fxv

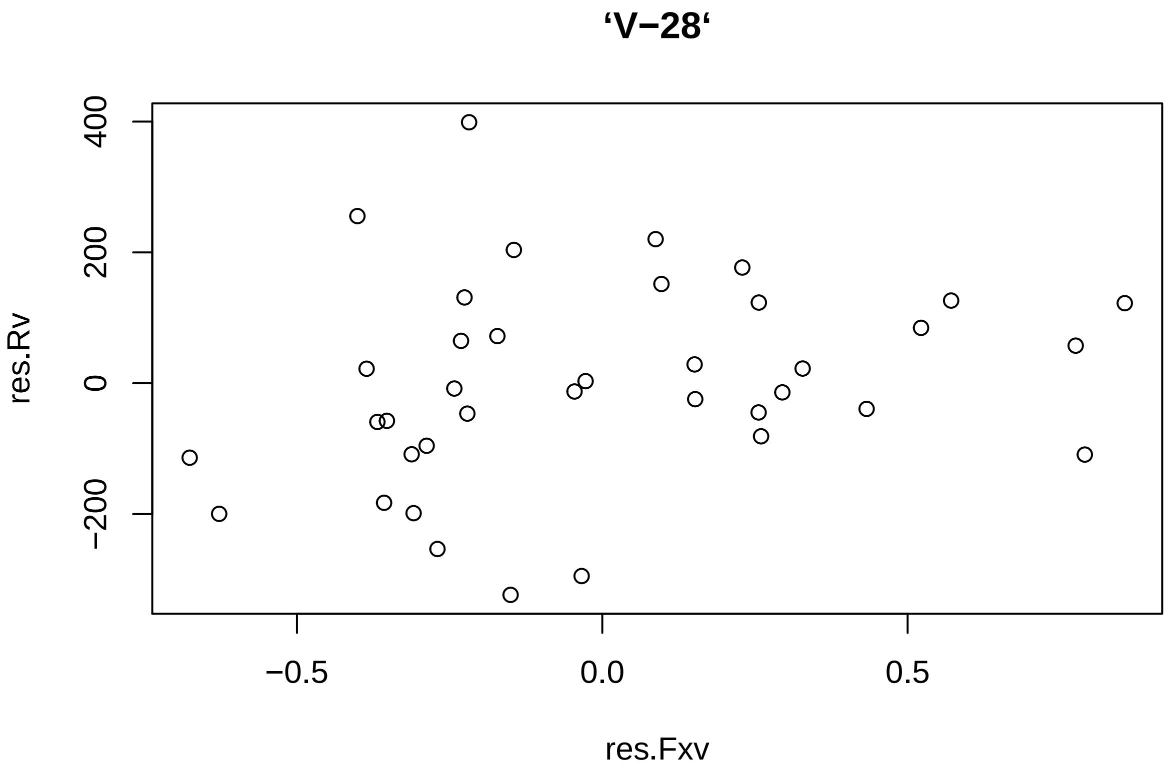

‘V−18‘
res.Rv
300
100
−100
−300
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
res.Fxv

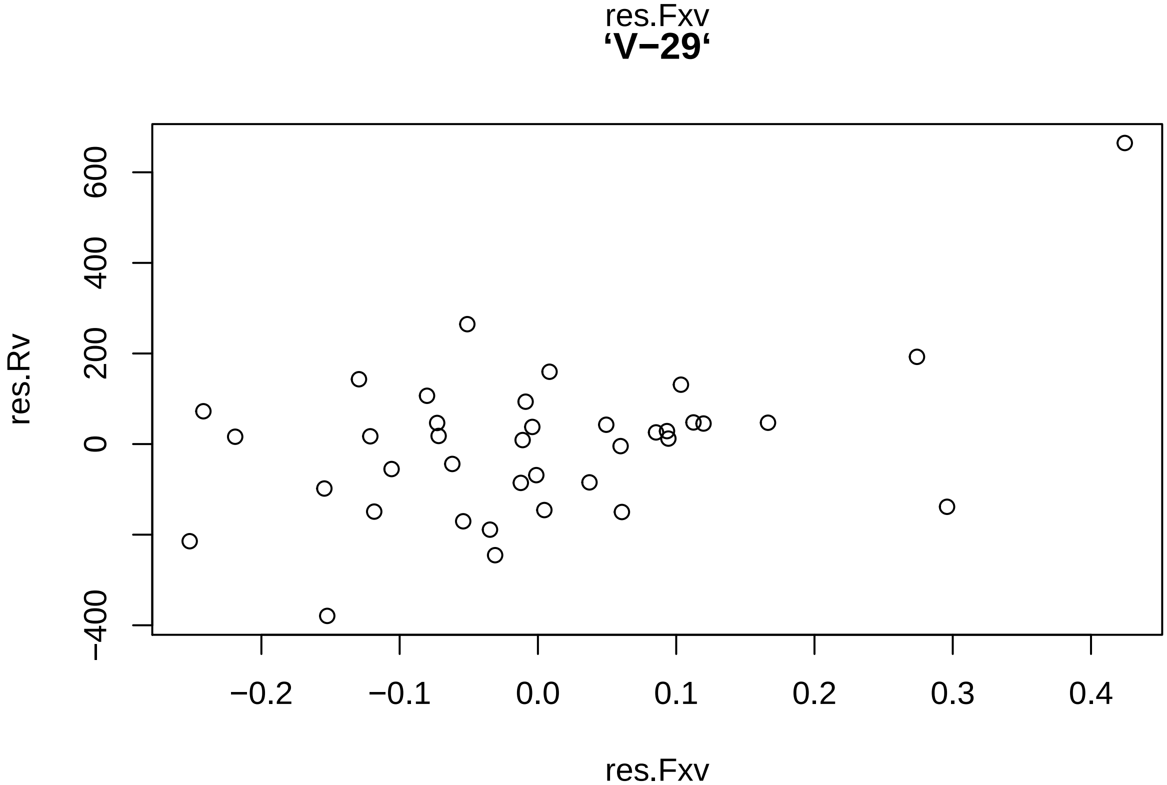

'V−20'
res.Rv
400
200
0
−200
−1.0
−0.5
0.0
0.5
res.Fxv


'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
res.Fxv
400
200
0
−200
−0.6
−0.4
−0.2
0.0
0.2
0.4


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






```
#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.
```

```r
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
```

```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] 2.063899
```

```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] 0.9995159
```

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

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

```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:
```

```r
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
```

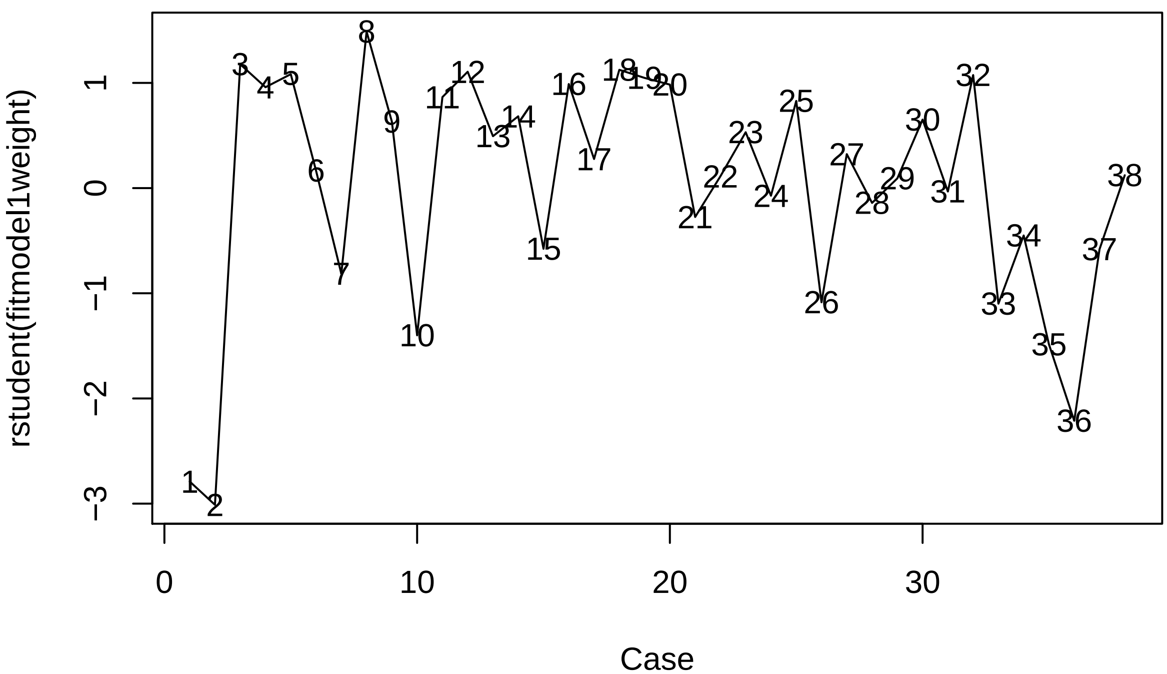
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel1-p1-1)   # find the catical value
```

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

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

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

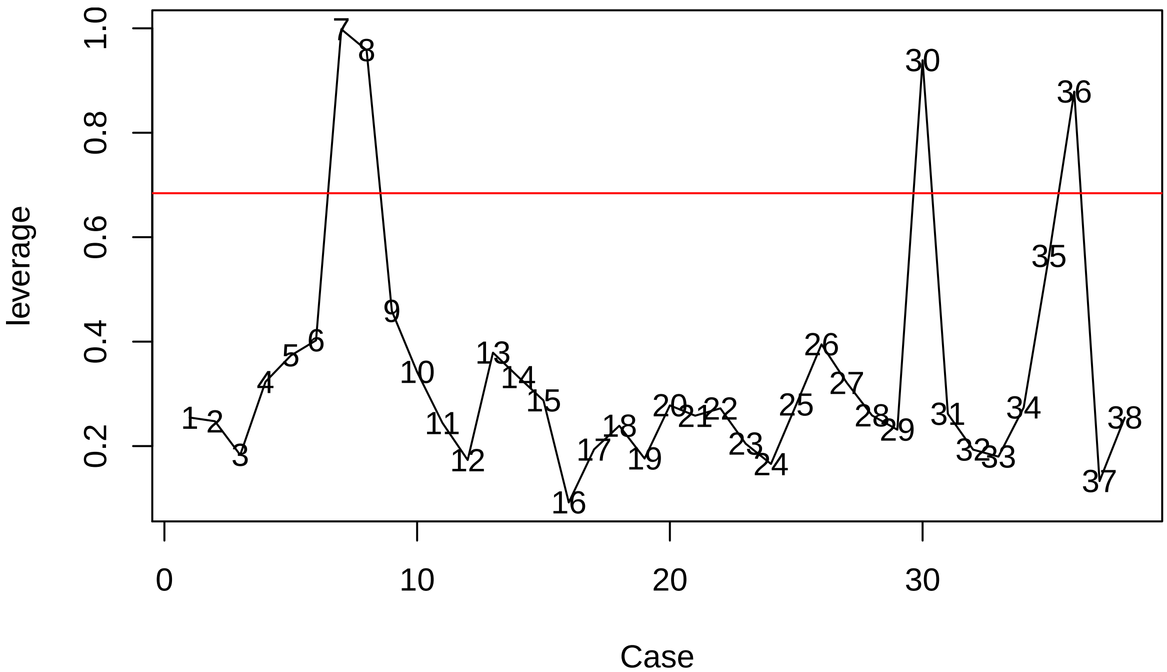
```r
#y outlier for model1
Case <- c(1:nmodel1)
plot(Case, rstudent(fitmodel1weight), type="l")
text(Case, rstudent(fitmodel1weight), Case)
```




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

```r
#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)
```

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




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

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

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

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

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

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

```r
# 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-
```

```r
# 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
```

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

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

```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] 0.9919199
```

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

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

```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] 2.045055
```

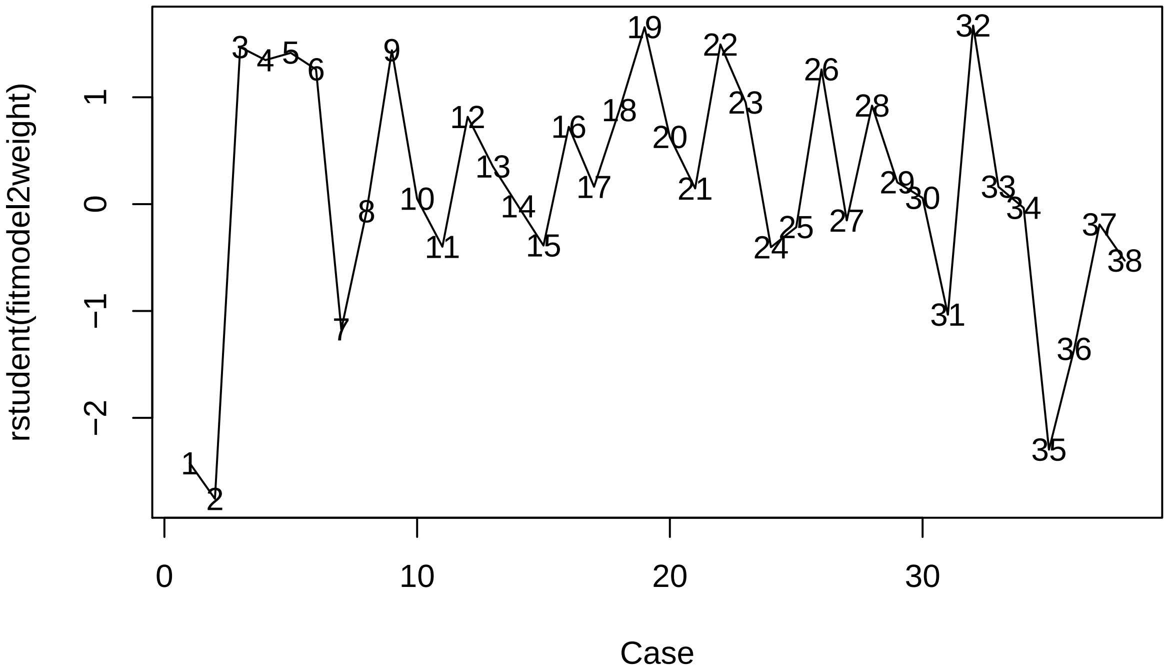
```r
#6 Once you obtain this value, you can compare it to the critical value for any given alpha level to de
# or you can find its P-value.
alpha <- 0.05
qt(1-alpha/2, nmodel2-17)   # find the catical value
```

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

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

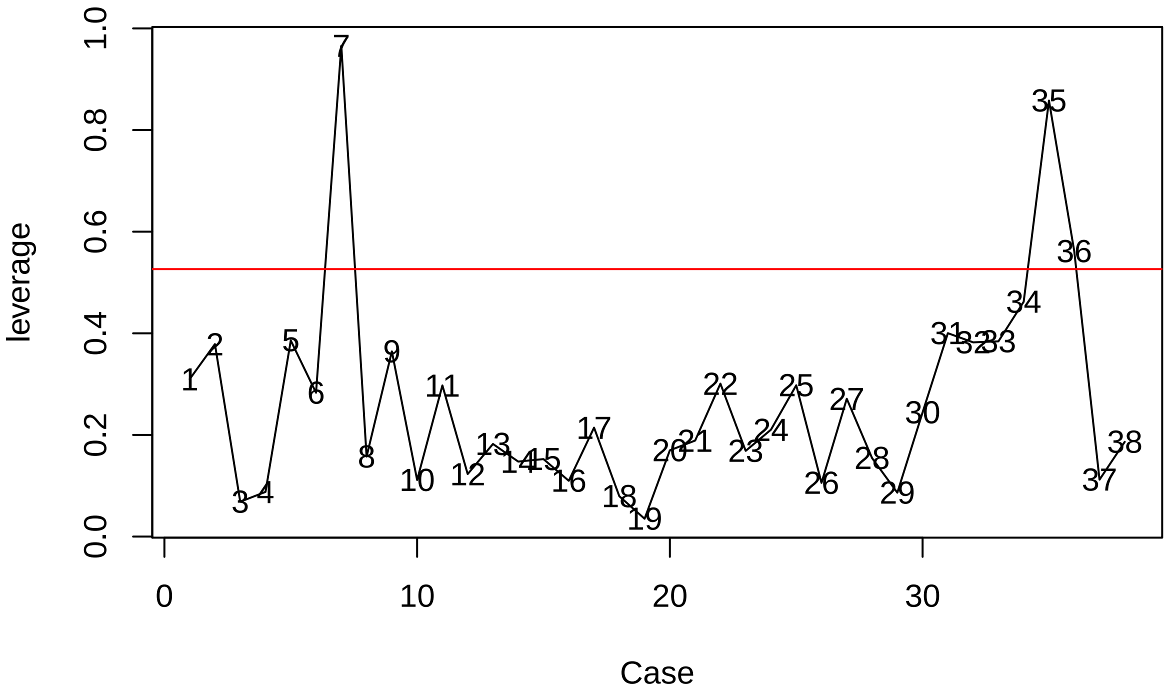
```
## [1] 0.05358353
```

```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
```