I was interning for an export/import firm. I used regression to predict the demand for their product using various global economic variables such as price of oil, USD conversion rate, taxes, shipping rates, cost of raw materials, etc. Since it was an export/import firm and not a manufacturing firm, they had to procure the product just in time. So for a large order, they would have to wait in order to procure the required quantity from different manufacturers. But however if they had the demand known in advance, they could assess and plan accordingly.

In [1]:
```
#specifying the url for data file
url <- "http://www.statsci.org/data/general/uscrime.txt"

#getting the data from the url
data <- read.table(url, header = T)
```

In [2]:
```
#for penalized regression -- it would be ideal to scale the data
#in order to ensure that all features are penalized equally.
#But in case of linear regression, it doesn't really matter.
#It will just shift the intercept and coefficients but the correlation re
```

In [3]:
```
#lets store the data point for which we need to predict
test <-data.frame(M = 14.0,So = 0, Ed = 10.0, Po1 = 12.0, Po2 = 15.5,LF =
                  M.F = 94.0, Pop = 150, NW = 1.1, U1 = 0.120, U2 = 3.6,
                  Ineq = 20.1, Prob = 0.040,Time = 39.0)
```

In [4]:
```
#printing the head of the scaled data.
head(data)
```

M	So	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth	Ineq	Prob	Time
15.1	1	9.1	5.8	5.6	0.510	95.0	33	30.1	0.108	4.1	3940	26.1	0.084602	26.2011
14.3	0	11.3	10.3	9.5	0.583	101.2	13	10.2	0.096	3.6	5570	19.4	0.029599	25.2999
14.2	1	8.9	4.5	4.4	0.533	96.9	18	21.9	0.094	3.3	3180	25.0	0.083401	24.3006
13.6	0	12.1	14.9	14.1	0.577	99.4	157	8.0	0.102	3.9	6730	16.7	0.015801	29.9012
14.1	0	12.1	10.9	10.1	0.591	98.5	18	3.0	0.091	2.0	5780	17.4	0.041399	21.2998
12.1	0	11.0	11.8	11.5	0.547	96.4	25	4.4	0.084	2.9	6890	12.6	0.034201	20.9995

In [5]:
```
#performing simple linear regression using all features

simple_model <- lm(Crime ~ ., data = data)

#printing the summary
summary(simple_model)
```

```
Call:
lm(formula = Crime ~ ., data = data)

Residuals:
    Min      1Q  Median      3Q     Max 
-395.74  -98.09   -6.69  112.99  512.67 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
M            8.783e+01  4.171e+01   2.106 0.043443 *  
So          -3.803e+00  1.488e+02  -0.026 0.979765    
Ed           1.883e+02  6.209e+01   3.033 0.004861 ** 
Po1          1.928e+02  1.061e+02   1.817 0.078892 .  
Po2         -1.094e+02  1.175e+02  -0.931 0.358830    
LF          -6.638e+02  1.470e+03  -0.452 0.654654    
M.F          1.741e+01  2.035e+01   0.855 0.398995    
Pop         -7.330e-01  1.290e+00  -0.568 0.573845    
NW           4.204e+00  6.481e+00   0.649 0.521279    
U1          -5.827e+03  4.210e+03  -1.384 0.176238    
U2           1.678e+02  8.234e+01   2.038 0.050161 .  
Wealth       9.617e-02  1.037e-01   0.928 0.360754    
Ineq         7.067e+01  2.272e+01   3.111 0.003983 ** 
Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *  
Time        -3.479e+00  7.165e+00  -0.486 0.630708    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 209.1 on 31 degrees of freedom
Multiple R-squared:  0.8031,	Adjusted R-squared:  0.7078 
F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

This was just a simple model. As per the summary, the most significant variables are M, Ed, Po1, Po2, U2, Prob and Ineq. Using this as a final model will definitely be a bad choice. Since we haven't performed any kind of feature selection or evaluated the model quality. The above model has an adjusted R-squared of 0.7078.

In [6]:
```
#Even though we know that the model is not good, let's use it to predict
pred1 <- predict(simple_model, test)
```

In [7]:
```
pred1
```

1: 155.434896887448

In [8]:
```
range(data$Crime)
```

342 1993

The above prediction doesn't make sense. The range of Crime in our data is [342, 1993] but our predicted value is 155.43 well below the lower range. Hence some overfitting has occurred.

In [9]:
```
#using features based on p-values is not recommended. So hence i resorted
#to some other complex methods.
```

Changing seed values leads to different results. I'm guessing this is due to the fact that our data is small. Please comment in the assessment if you think there's a different reason.

In [10]:
```
#Let's move on to some feature selection and cross validation.

#I have used recursive feature selection to identify the important ones.

#using repeatedcv

#setting the seed in order to have consistent results.

set.seed(42)
library(caret)

subsets <- c(1:15)

ctrl <- rfeControl(functions = lmFuncs,
                   method = "repeatedcv", number = 10,
                   repeats = 20,
                   verbose = FALSE)
lmProfile <- rfe(data[,-16], data[,16],
                 sizes = subsets,
                 rfeControl = ctrl)
lmProfile
```

```
Warning message:
“package ‘caret’ was built under R version 3.4.4”Loading required pack
age: lattice
Loading required package: ggplot2
Warning message:
“package ‘ggplot2’ was built under R version 3.4.4”Warning message in
```

```
as.POSIXlt.POSIXct(Sys.time()):
“unknown timezone 'zone/tz/2018e.1.0/zoneinfo/America/New_York'”


Recursive feature selection

Outer resampling method: Cross-Validated (10 fold, repeated 20 times)

Resampling performance over subset size:

 Variables  RMSE Rsquared   MAE RMSESD RsquaredSD  MAESD Selected
         1 360.5   0.3647 292.8 131.18     0.3076 103.81
         2 348.4   0.3617 279.3 133.15     0.2930 111.22
         3 350.6   0.3344 283.3 129.07     0.2873 108.87
         4 336.1   0.4167 280.2 100.48     0.3107  90.42
         5 317.4   0.4940 261.9 102.82     0.3165  97.17
         6 304.1   0.5281 253.8 100.19     0.3079  93.73
         7 307.8   0.5245 256.1  99.79     0.3048  90.85
         8 275.6   0.5818 227.4  90.54     0.2904  77.23
         9 241.7   0.6461 194.1  95.80     0.2782  78.77
        10 228.5   0.6721 185.5  85.79     0.2579  69.70        *
        11 238.3   0.6307 193.9  93.33     0.2753  74.14
        12 251.1   0.5939 203.9  95.64     0.2941  79.27
        13 253.8   0.5910 206.5  98.01     0.2932  81.27
        14 258.8   0.5861 209.8 101.27     0.2939  85.52
        15 266.6   0.5817 217.4  96.66     0.2822  84.21

The top 5 variables (out of 10):
   U1, Prob, LF, Po1, Ed
```

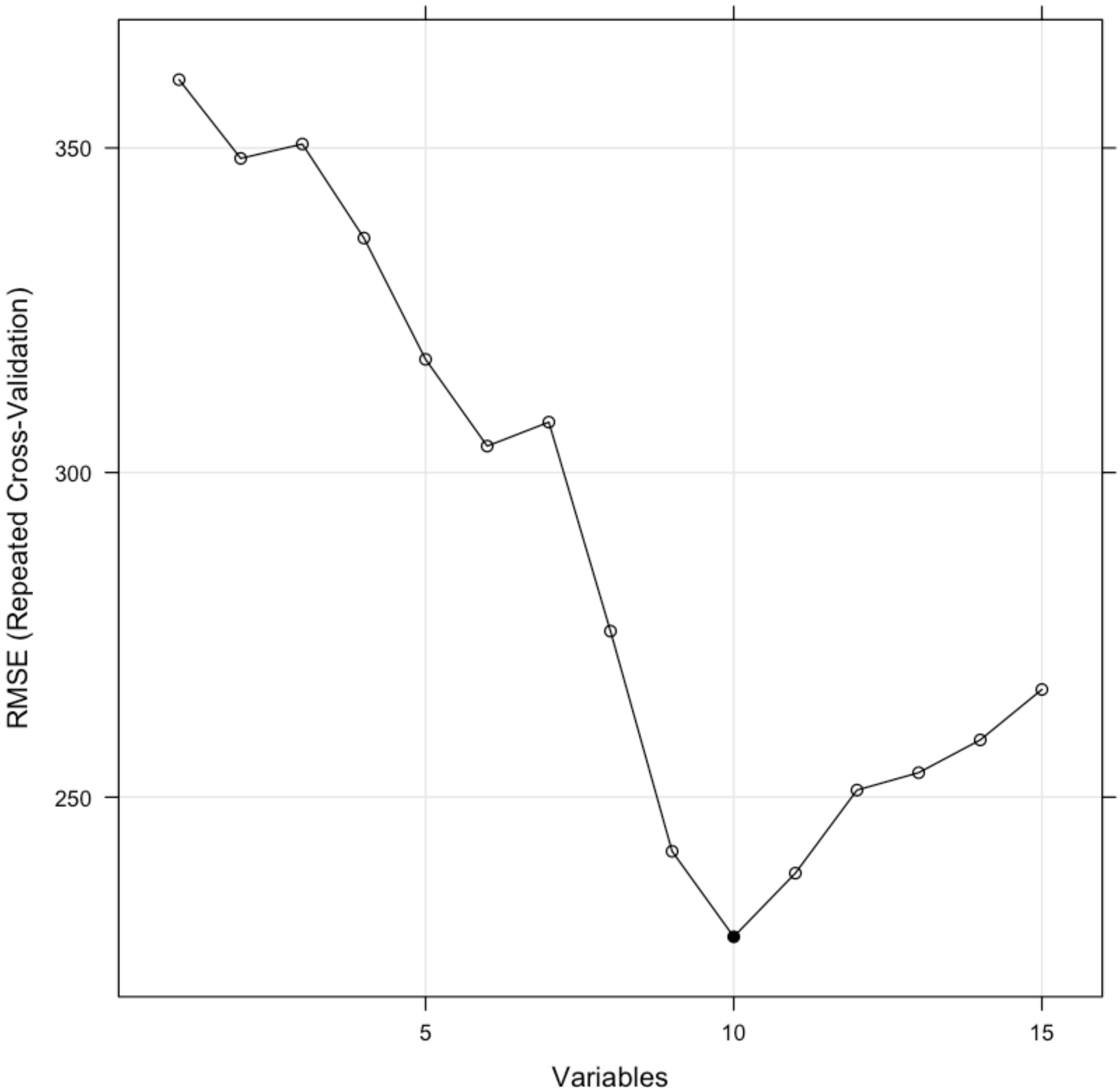
In [11]:

```
#the model above picks 10 features and those are as follows.
predictors(lmProfile)
```

'U1' 'Prob' 'LF' 'Po1' 'Ed' 'U2' 'Po2' 'M' 'Ineq' 'So'

It can be seen that the lowest RMSE is achieved at 10 variables.

In [12]:
```
trellis.par.set(caretTheme())
plot(lmProfile, type = c("g", "o"))
```



This is the model using the best 10 features selected above.

In [13]:
```
lmProfile$fit
```

```
Call:
lm(formula = y ~ ., data = tmp)

Coefficients:
(Intercept)           U1         Prob           LF          Po1
Ed
   -5099.78     -2925.15     -4000.57       531.84       177.49
210.85
         U2          Po2            M         Ineq           So
     150.25       -79.51       100.88        58.80        78.48
```

This prediction is reasonable and well within the bounds of Crime in our dataset.

In [14]:
```
predict(lmProfile, test)
```

1: 870.683361434228

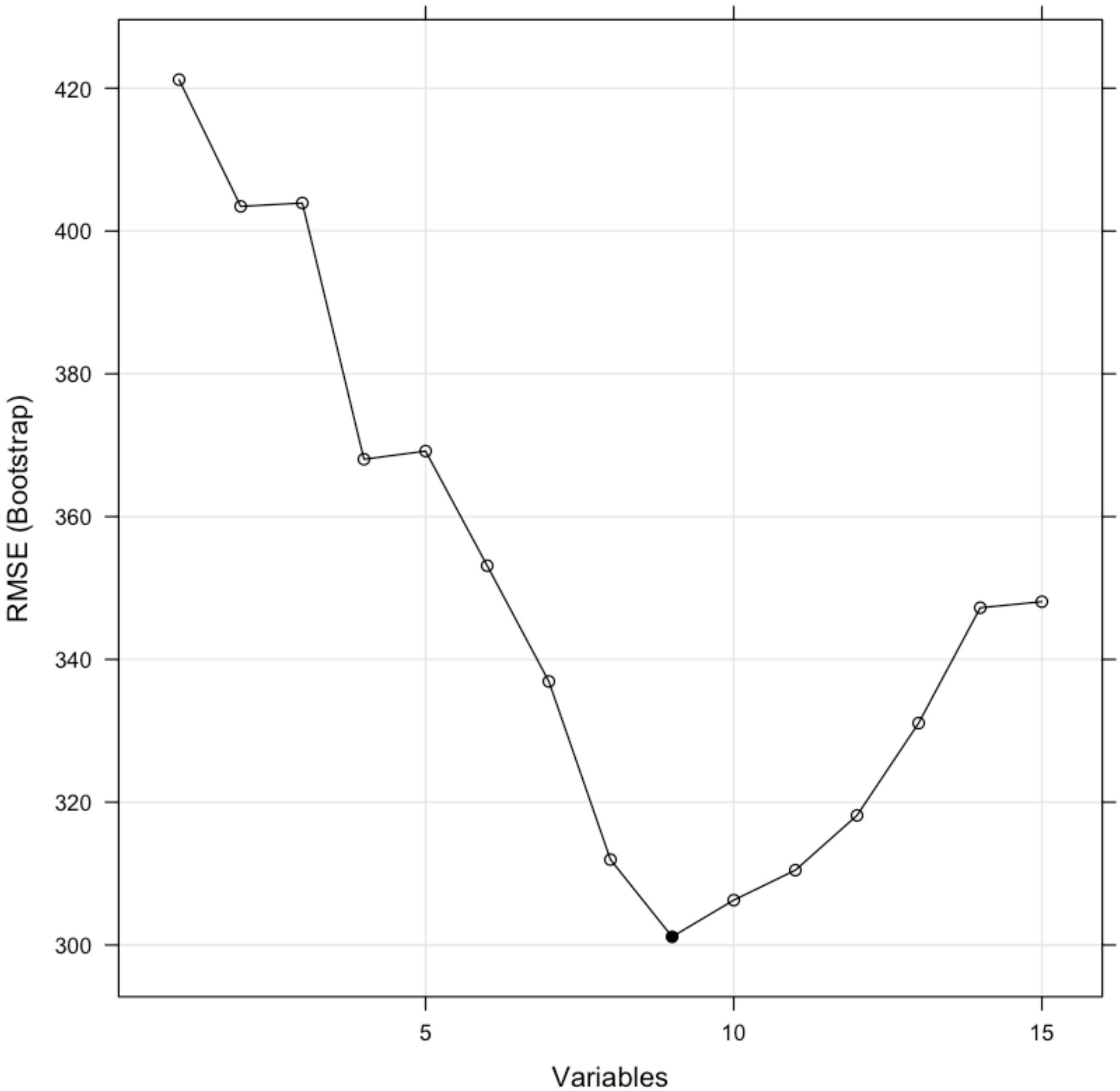
In [15]:
```
#using simple cross-validation instead of repeatedcv

library(caret)

subsets <- c(1:15)

ctrl <- rfeControl(functions = lmFuncs,
                   number = 10,
                   verbose = FALSE)
lmProfile <- rfe(data[,-16], data[,16],
                 sizes = subsets,
                 rfeControl = ctrl)
lmProfile
```

```
Recursive feature selection

Outer resampling method: Bootstrapped (10 reps)

Resampling performance over subset size:

 Variables  RMSE Rsquared   MAE RMSESD RsquaredSD MAESD Selected
         1 421.2   0.1625 327.9  66.55    0.16540 51.06         
         2 403.5   0.2513 314.6  85.45    0.18199 65.04         
         3 403.9   0.1843 315.1 100.31    0.19749 81.52         
         4 368.0   0.3063 300.0  97.03    0.20009 78.32         
         5 369.2   0.3564 301.1 102.23    0.21299 81.21         
         6 353.1   0.4203 286.0  73.72    0.17353 65.57         
         7 336.9   0.5009 269.0  58.94    0.12389 47.85         
         8 312.0   0.5579 250.0  61.89    0.11945 47.63         
         9 301.2   0.5823 233.3  53.05    0.11089 43.73        *
        10 306.3   0.5930 240.2  72.98    0.09644 61.24         
        11 310.5   0.5825 243.4  71.01    0.10655 52.77         
        12 318.1   0.5637 244.9  62.75    0.10510 45.30         
        13 331.1   0.5242 253.1  70.98    0.14008 47.73         
        14 347.2   0.5052 264.4  74.01    0.12925 49.46         
        15 348.1   0.5090 268.4  73.45    0.13306 50.30         

The top 5 variables (out of 9):
   Prob, U1, LF, Po1, Ed
```

In [16]:
```
###So now 9 features are selected. Lets see what they are.
predictors(lmProfile)

#### features from repeatedcv - U1' 'Prob' 'LF' 'Po1' 'Ed' 'U2' 'Po2' 'M'
```

'Prob' 'U1' 'LF' 'Po1' 'Ed' 'U2' 'Po2' 'So' 'M'

In [17]:
```
trellis.par.set(caretTheme())
plot(lmProfile, type = c("g", "o"))
```



In [18]:
```
#All the features are common. Hence our results are consistent.
```

In [19]:
```
lmProfile$fit
```

```
Call:
lm(formula = y ~ ., data = tmp)

Coefficients:
(Intercept)         Prob           U1           LF          Po1
Ed
    -3979.6      -4289.1      -1632.0       1940.5        193.1
117.9
         U2          Po2           So            M
      143.1       -116.7        293.7        116.7
```

In [20]:
```
predict(lmProfile, test)
```

1: 730.522879940032

In [21]:
```
#using caret package to perform cross validation on simple model.

#leave one out cross validation is performed.

train.control <- trainControl(method = "LOOCV")
model <- train(Crime ~., data = data, method = "lm",
               trControl = train.control)
print(model)
```

```
Linear Regression

47 samples
15 predictors

No pre-processing
Resampling: Leave-One-Out Cross-Validation
Summary of sample sizes: 46, 46, 46, 46, 46, 46, ...
Resampling results:

  RMSE     Rsquared   MAE
  274.424  0.5265155  209.0678

Tuning parameter 'intercept' was held constant at a value of TRUE
```

In [22]:
```
predict(model, test)
```

1: 155.434896887448

It can be seen that earlier simple model gave Rsquared of 0.8031 and now the cross validated Rsquared is 0.5265. Hence, it can be said that the simple model is overfitting and will not be able to generalize well.

In [23]:
```
train.control <- trainControl(method = "repeatedcv", repeats = 10)
model <- train(Crime ~., data = data, method = "lm",
               trControl = train.control)
print(model)
```

```
Linear Regression

47 samples
15 predictors

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 10 times)
Summary of sample sizes: 43, 41, 42, 42, 44, 43, ...
Resampling results:

  RMSE      Rsquared   MAE
  263.0553  0.5930593  212.3239

Tuning parameter 'intercept' was held constant at a value of TRUE
```

In [24]:
```
#using the features we got from rfe (recursive feature selection)

train.control <- trainControl(method = "repeatedcv", repeats = 5, number
model <- train(Crime~U1+Prob+LF+Po1+Ed+U2+Po2+M+Ineq+So, data = data, met
               trControl = train.control)
print(model)
```

```
Linear Regression

47 samples
10 predictors

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 42, 41, 42, 42, 42, 43, ...
Resampling results:

  RMSE      Rsquared   MAE
  225.9265  0.6796605  185.271

Tuning parameter 'intercept' was held constant at a value of TRUE
```

The above model was able to achieve highest cross validated Rsquared and lowest RMSE till now.

In [26]:

```
#Lets use it to predict.
predict(model, test)
```

1: 870.683361434228

In [27]:

```
#now using the step function for feature selection -- this is based on AI
#starting with a simple model (in case of forward), we add features that
#the best model is one with lowest AIC.

min.model = lm(Crime ~ 1, data= data)
biggest <- formula(lm(Crime~., data))

model_step_forward <- step(min.model , scope = biggest, scale = 0,
     direction = "forward",
     trace = 1, keep = NULL, steps = 1000, k = 2)
```

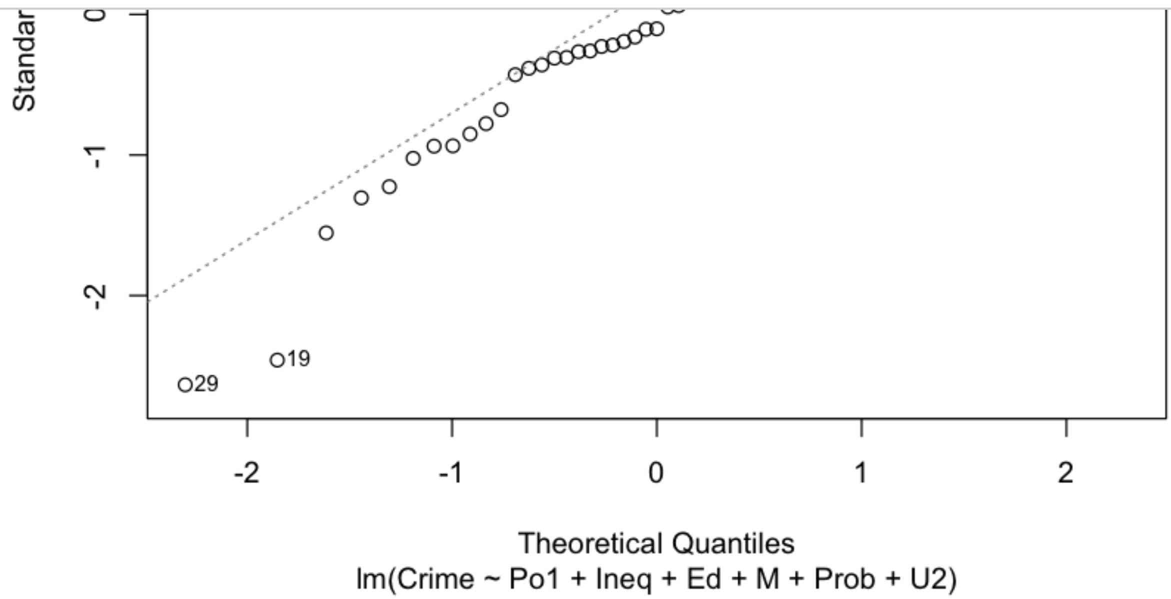
```
<none>                  1803290 508.08
+ U1        1     52313 1750977 508.70
+ Pop       1     47719 1755571 508.82
+ Po2       1     37967 1765323 509.08
+ So        1     21971 1781320 509.51
+ Time      1     10194 1793096 509.82
+ LF        1       990 1802301 510.06
+ NW        1       797 1802493 510.06

Step:  AIC=504.79
Crime ~ Po1 + Ineq + Ed + M + Prob + U2

         Df Sum of Sq     RSS    AIC
<none>                  1611057 504.79
+ Wealth    1     59910 1551147 505.00
+ U1        1     54830 1556227 505.16
+ Pop       1     51320 1559737 505.26
+ M.F       1     30945 1580112 505.87
+ Po2       1     25017 1586040 506.05
+ So        1     17958 1593098 506.26
```

In [28]:

```
# Crime ~ Po1 + Ineq + Ed + M + Prob + U2 (best model based on forward st
predict(model_step_forward, test)
```

1: 1304.24521072363

In [29]:
```
#it can be seen from the QQ plots of residuals that the normality assumpt
#errors is violated or may be don't have enough data point.
plot(model_step_forward)
```



In [30]:
```
#now lets try backward step feature selection.
max.model = lm(Crime ~ ., data= data)
smallest <- formula(lm(Crime~1, data))

model_step_backward <- step(max.model , scope = smallest, scale = 0,
     direction = "backward",
     trace = 1, keep = NULL, steps = 1000, k = 2)
```

```
- Wealth    1     26493 1453068 503.93
<none>                  1426575 505.07
- M.F       1     84491 1511065 505.77
- U1        1     99463 1526037 506.24
- Prob      1    198571 1625145 509.20
- U2        1    208880 1635455 509.49
- M         1    320926 1747501 512.61
- Ed        1    386773 1813348 514.35
- Ineq      1    594779 2021354 519.45
- Po1       1   1127277 2553852 530.44

Step:  AIC=503.93
Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob

       Df Sum of Sq     RSS    AIC
<none>              1453068 503.93
- M.F   1    103159 1556227 505.16
- U1    1    127044 1580112 505.87
- Prob  1    247978 1701046 509.34
- U2    1    255443 1708511 509.55
```

In [31]:
```
#model selected -- Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob
```

Interesting thing to note is that some of the features have very strong predictive power and hence they are selected by all feature techniques. This sort of validates all our methods and gives confidence in our results.

In [32]:
```
#prediction from the above backward model.
predict(model_step_backward, test)
```

1: 1038.41335997717

In [ ]: