# Homework 4 - Week 4

## Question 9.1

***Using the same crime data set uscrime.txt as in Question 8.2, apply Principal Component Analysis and then create a regression model using the first few principal components. Specify your new model in terms of the original variables (not the principal components), and compare its quality to that of your solution to Question 8.2. You can use the R function prcomp for PCA. (Note that to first scale the data, you can include scale. = TRUE to scale as part of the PCA function. Don't forget that, to make a prediction for the new city, you'll need to unscale the coefficients (i.e., do the scaling calculation in reverse)!)***

In [2]:
```
library(dplyr)
```

In [6]:
```
# install.packages("DAAG")
library(DAAG)
```

In [11]:
```
dat <- read.table("uscrime.txt", header = TRUE)
```

In [13]:
```
head(dat)
```

| M | So | Ed | Po1 | Po2 | LF | M.F | Pop | NW | U1 | U2 | Wealth | Ineq | Prob | Time | Crime |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 791 |
| 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 | 1635 |
| 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 | 578 |
| 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 | 1969 |
| 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 | 1234 |
| 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 | 682 |

In [14]:
```
str(dat)
```

```
'data.frame':	47 obs. of  16 variables:
 $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
 $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
 $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
 $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
 $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
 $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...
 $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
 $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
 $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
 $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...
 $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
 $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
 $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
 $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
 $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
 $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

We call prcomp() to do PCA on the data.

The goal is to draw a graph that shows how the samples are related (or not related) to each other.

In [15]:
```
my.prc <- prcomp(dat[,-16], center=TRUE, scale=TRUE)
```

In [16]:
```
summary(my.prc)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
Standard deviation     2.4534 1.6739 1.4160 1.07806 0.97893 0.74377 0.56729
Proportion of Variance 0.4013 0.1868 0.1337 0.07748 0.06389 0.03688 0.02145
Cumulative Proportion  0.4013 0.5880 0.7217 0.79920 0.86308 0.89996 0.92142
                           PC8     PC9    PC10    PC11    PC12    PC13   PC14
Standard deviation     0.55444 0.48493 0.44708 0.41915 0.35804 0.26333 0.2418
Proportion of Variance 0.02049 0.01568 0.01333 0.01171 0.00855 0.00462 0.0039
Cumulative Proportion  0.94191 0.95759 0.97091 0.98263 0.99117 0.99579 0.9997
                          PC15
Standard deviation     0.06793
Proportion of Variance 0.00031
Cumulative Proportion  1.00000
```

In [17]:
```
my.prc$x
```

| PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 | PC8 | PC9 | |
|---|---|---|---|---|---|---|---|---|---|
| -4.1992835 | -1.09383120 | -1.11907395 | 0.67178115 | 0.055283376 | 0.30733835 | -0.566408161 | -0.007801727 | 0.223509947 | 0.45 |
| 1.1726630 | 0.67701360 | -0.05244634 | -0.08350709 | -1.173199821 | -0.58323731 | 0.195611187 | 0.154566472 | 0.436777195 | 0.21 |
| -4.1737248 | 0.27677501 | -0.37107658 | 0.37793995 | 0.541345246 | 0.71872230 | 0.103306929 | 0.351138883 | 0.062992321 | -0.06 |
| 3.8349617 | -2.57690596 | 0.22793998 | 0.38262331 | -1.644746496 | 0.72948841 | 0.266994985 | -1.547460841 | -0.379541806 | 0.22 |
| 1.8392999 | 1.33098564 | 1.27882805 | 0.71814305 | 0.041590320 | -0.39409015 | 0.070507664 | -0.543237437 | 0.224632448 | 0.47 |
| 2.9072336 | -0.33054213 | 0.53288181 | 1.22140635 | 1.374360960 | -0.69225131 | 0.226482092 | 0.562323186 | 0.417722172 | 0.09 |
| 0.2457752 | -0.07362562 | -0.90742064 | 1.13685873 | 0.718644387 | -0.93107472 | 0.307507661 | 1.056861503 | -1.160218292 | 0.79 |
| -0.1301330 | -1.35985577 | 0.59753132 | 1.44045387 | -0.222781388 | 0.04912052 | 0.911404993 | 0.693339330 | -0.421314146 | 0.61 |
| -3.6103169 | -0.68621008 | 1.28372246 | 0.55171150 | -0.324292990 | 0.12683417 | -0.417420968 | -0.053270500 | 0.232662026 | 0.06 |
| 1.1672376 | 3.03207033 | 0.37984502 | -0.28887026 | -0.646056610 | 0.33130781 | 0.009579488 | -0.329270845 | -0.123629746 | 0.20 |
| 2.5384879 | -2.66771358 | 1.54424656 | -0.87671210 | -0.324083561 | 0.44365740 | -0.182961180 | 0.587179568 | -0.070907596 | -0.55 |
| 1.0065920 | -0.06044849 | 1.18861346 | -1.31261964 | 0.358087724 | 0.25696957 | -0.462577031 | 0.307351101 | -0.105197263 | -0.13 |
| 0.5161143 | 0.97485189 | 1.83351610 | -1.59117618 | 0.599881946 | 1.04761756 | -0.494631320 | 0.753702337 | -0.384056907 | -0.34 |
| 0.4265556 | 1.85044812 | 1.02893477 | -0.07789173 | 0.741887592 | 0.61569775 | -0.087093101 | -0.046931419 | -0.159138488 | 0.28 |
| -3.3435299 | 0.05182823 | -1.01358113 | 0.08840211 | 0.002969448 | 0.17074576 | 1.040213660 | -0.139392628 | -0.147546022 | -1.02 |
| -3.0310689 | -2.10295524 | -1.82993161 | 0.52347187 | -0.387454246 | -0.20965321 | 0.262430717 | 0.641818600 | 0.526895635 | 0.82 |
| -0.2262961 | 1.44939774 | -1.37565975 | 0.28960865 | 1.337784608 | -0.25633983 | -0.754882880 | -0.959968310 | 0.351808733 | -0.04 |
| -0.1127499 | -0.39407030 | -0.38836278 | 3.97985093 | 0.410914404 | 0.09317136 | -1.227238054 | 0.280226677 | -0.412734008 | -1.07 |
| 2.9195668 | -1.58646124 | 0.97612613 | 0.78629766 | 1.356288600 | -0.89044651 | 0.387161139 | -0.002276046 | 0.555855685 | 0.59 |
| 2.2998485 | -1.73396487 | -2.82423222 | -0.23281758 | -0.653038858 | 0.68615337 | -0.401936004 | 0.240456772 | 0.341543809 | 0.22 |
| 1.1501667 | 0.13531015 | 0.28506743 | -2.19770548 | 0.084621572 | 0.45958300 | -0.179283176 | 0.772072202 | -0.344317021 | -0.19 |
| -5.6594827 | -1.09730404 | 0.10043541 | -0.05245484 | -0.689327990 | 0.13338054 | -1.337728458 | 0.261648468 | 0.225568667 | 0.36 |
| -0.1011749 | -0.57911362 | 0.71128354 | -0.44394773 | 0.689939865 | 0.54002731 | 0.995827754 | 0.371597176 | 1.073655584 | 0.03 |
| 1.3836281 | 1.95052341 | -2.98485490 | -0.35942784 | -0.744371276 | 0.01453851 | 0.042135169 | -0.210603749 | -0.111463892 | 0.57 |
| 0.2727756 | 2.63013778 | 1.83189535 | 0.05207518 | 0.803692524 | 1.52313508 | -0.341012092 | 0.390172476 | -0.015090214 | -0.10 |
| 4.0565577 | 1.17534729 | -0.81690756 | 1.66990720 | -2.895110075 | -0.47766314 | -0.110906098 | 0.991890307 | 0.232407672 | -0.72 |
| 0.8929694 | 0.79236692 | 1.26822542 | -0.57575615 | 1.830793964 | -1.11656766 | -0.199196211 | -0.044269305 | -0.015729946 | -0.04 |
| 0.1514495 | 1.44873320 | 0.10857670 | -0.51040146 | -1.023229895 | -0.74149513 | 0.113082804 | -0.677219677 | 0.151930973 | 0.07 |
| 3.5592481 | -4.76202163 | 0.75080576 | 0.64692974 | 0.309946510 | 0.72486153 | 0.248081636 | -0.844089307 | 0.230269486 | -0.34 |
| -4.1184576 | -0.38073981 | 1.43463965 | 0.63330834 | -0.254715638 | -0.42316550 | -0.116127247 | -0.891169193 | -0.011731985 | -0.43 |
| -0.6811731 | 1.66926027 | -2.88645794 | -1.30977099 | -0.470913997 | -0.45866080 | 0.704852096 | -0.538600585 | 0.439137868 | -0.70 |
| 1.7157269 | -1.30836339 | -0.55971313 | -0.70557980 | 0.331277622 | 1.30802615 | -0.786980332 | -0.067086938 | -0.169888285 | 0.07 |
| -1.8860627 | 0.59058174 | 1.43570145 | 0.18239089 | 0.291863659 | -0.13885903 | 0.767856496 | 0.027448832 | -0.773125607 | 0.12 |
| 1.9526349 | 0.52395429 | -0.75642216 | 0.44289927 | 0.723474420 | -0.42036754 | 0.181257930 | 0.115379461 | -0.101718594 | 0.32 |
| 1.5888864 | -3.12998571 | -1.73107199 | -1.68604766 | 0.665406182 | 0.54144206 | -0.449541256 | -0.276891496 | 0.007657702 | 0.20 |
| 1.0709414 | -1.65628271 | 0.79436888 | -1.85172698 | 0.020031154 | -2.43356674 | -0.333843509 | 0.384707595 | 0.642612190 | -0.72 |
| -4.1101715 | 0.15766712 | 2.36296974 | -0.56868399 | -2.469679496 | 0.07239996 | -0.343611407 | 0.157984131 | 0.915881371 | 0.48 |
| -0.7254706 | 2.89263339 | -0.36348376 | -0.50612576 | 0.028157162 | 1.06465126 | 0.863051754 | -0.058247210 | 0.341385143 | -0.13 |
| -3.3451254 | -0.95045293 | 0.19551398 | -0.27716645 | 0.487259213 | -0.20571166 | 0.966860079 | 0.059557654 | 0.039345212 | 0.03 |
| -1.0644466 | -1.05265304 | 0.82886286 | -0.12042931 | -0.645884788 | 0.63320546 | 0.767470212 | -0.704833575 | -1.109887730 | 0.10 |
| 1.4933989 | 1.86712106 | 1.81853582 | -1.06112429 | 0.009855774 | -1.03480444 | -0.589160590 | -0.468876595 | -0.528478950 | 0.43 |
| -0.6789284 | 1.83156328 | -1.65435992 | 0.95121379 | 2.115630145 | -0.02332805 | -0.557413301 | -0.963360913 | 0.485515025 | 0.00 |
| -2.4164258 | -0.46701087 | 1.42808323 | 0.41149015 | -0.867397522 | -1.13982198 | 0.041128192 | -0.573696577 | -0.773992630 | -0.44 |
| 2.2978729 | 0.41865689 | -0.64422929 | -0.63462770 | -0.703116983 | -0.65215040 | -0.442990964 | -0.093002011 | -0.515838387 | 0.24 |
| -2.9245282 | -1.19488555 | -3.35139309 | -1.48966984 | 0.806659622 | -0.48157983 | 0.233636019 | 0.379908278 | -0.815127937 | -0.54 |
| 1.7654525 | 0.95655926 | 0.98576138 | 1.05683769 | 0.542466034 | 0.71712602 | 0.847914876 | 0.172381544 | 0.657987377 | -0.48 |
| 2.3125056 | 2.56161119 | -1.58223354 | 0.59863946 | -1.140712406 | 0.39563373 | -0.171412192 | 0.327844331 | -0.167078790 | -0.00 |

In [18]:
```
my.prc$rotation
```

| | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 | PC8 | PC |
|---|---|---|---|---|---|---|---|---|---|
| **M** | -0.30371194 | 0.06280357 | 0.1724199946 | -0.02035537 | -0.35832737 | -0.449132706 | -0.15707378 | -0.55367691 | 0.1547479 |
| **So** | -0.33088129 | -0.15837219 | 0.0155433104 | 0.29247181 | -0.12061130 | -0.100500743 | 0.19649727 | 0.22734157 | -0.6559987 |
| **Ed** | 0.33962148 | 0.21461152 | 0.0677396249 | 0.07974375 | -0.02442839 | -0.008571367 | -0.23943629 | -0.14644678 | -0.4432697 |
| **Po1** | 0.30863412 | -0.26981761 | 0.0506458161 | 0.33325059 | -0.23527680 | -0.095776709 | 0.08011735 | 0.04613156 | 0.1942547 |
| **Po2** | 0.31099285 | -0.26396300 | 0.0530651173 | 0.35192809 | -0.20473383 | -0.119524780 | 0.09518288 | 0.03168720 | 0.1951207 |
| **LF** | 0.17617757 | 0.31943042 | 0.2715301768 | -0.14326529 | -0.39407588 | 0.504234275 | -0.15931612 | 0.25513777 | 0.1439349 |
| **M.F** | 0.11638221 | 0.39434428 | -0.2031621598 | 0.01048029 | -0.57877443 | -0.074501901 | 0.15548197 | -0.05507254 | -0.2437825 |
| **Pop** | 0.11307836 | -0.46723456 | 0.0770210971 | -0.03210513 | -0.08317034 | 0.547098563 | 0.09046187 | -0.59078221 | -0.2024483 |
| **NW** | -0.29358647 | -0.22801119 | 0.0788156621 | 0.23925971 | -0.36079387 | 0.051219538 | -0.31154195 | 0.20432828 | 0.1898417 |
| **U1** | 0.04050137 | 0.00807439 | -0.6590290980 | -0.18279096 | -0.13136873 | 0.017385981 | -0.17354115 | -0.20206312 | 0.0206934 |
| **U2** | 0.01812228 | -0.27971336 | -0.5785006293 | -0.06889312 | -0.13499487 | 0.048155286 | -0.07526787 | 0.24369650 | 0.0557601 |
| **Wealth** | 0.37970331 | -0.07718862 | 0.0100647664 | 0.11781752 | 0.01167683 | -0.154683104 | -0.14859424 | 0.08630649 | -0.2319669 |
| **Ineq** | -0.36579778 | -0.02752240 | -0.0002944563 | -0.08066612 | -0.21672823 | 0.272027031 | 0.37483032 | 0.07184018 | -0.0249438 |
| **Prob** | -0.25888661 | 0.15831708 | -0.1176726436 | 0.49303389 | 0.16562829 | 0.283535996 | -0.56159383 | -0.08598908 | -0.0530689 |
| **Time** | -0.02062867 | -0.38014836 | 0.2235664632 | -0.54059002 | -0.14764767 | -0.148203050 | -0.44199877 | 0.19507812 | -0.2355136 |

### rotation matrix is a formula that converts from the original variables to Principal Components

**So PC1 is nothing but (-0.30371194 * M) + (-0.33088129 * SO) + (0.33962148 * Ed) + ............ + (-0.02062867 * Time), similarly it goes for other PC's**

In [24]:
```
# We use **sdev** to calculate the variation in the original data each PC accounts for

var <- my.prc$sdev^2
```

In [25]:
```
var
```

6.01895265650583   2.80184702648299   2.0049443337507   1.16220780064104   0.958298971721856
0.55319389968655   0.321818687025561   0.307401270140502   0.23515529207527   0.19988093081482
0.175685402581489   0.128190107426395   0.0693416913365943   0.058467765048427   0.00461416476196681

In [26]:
```
var.per <- round(var/sum(var)*100, 1) # PERCENTAGE of variation
```

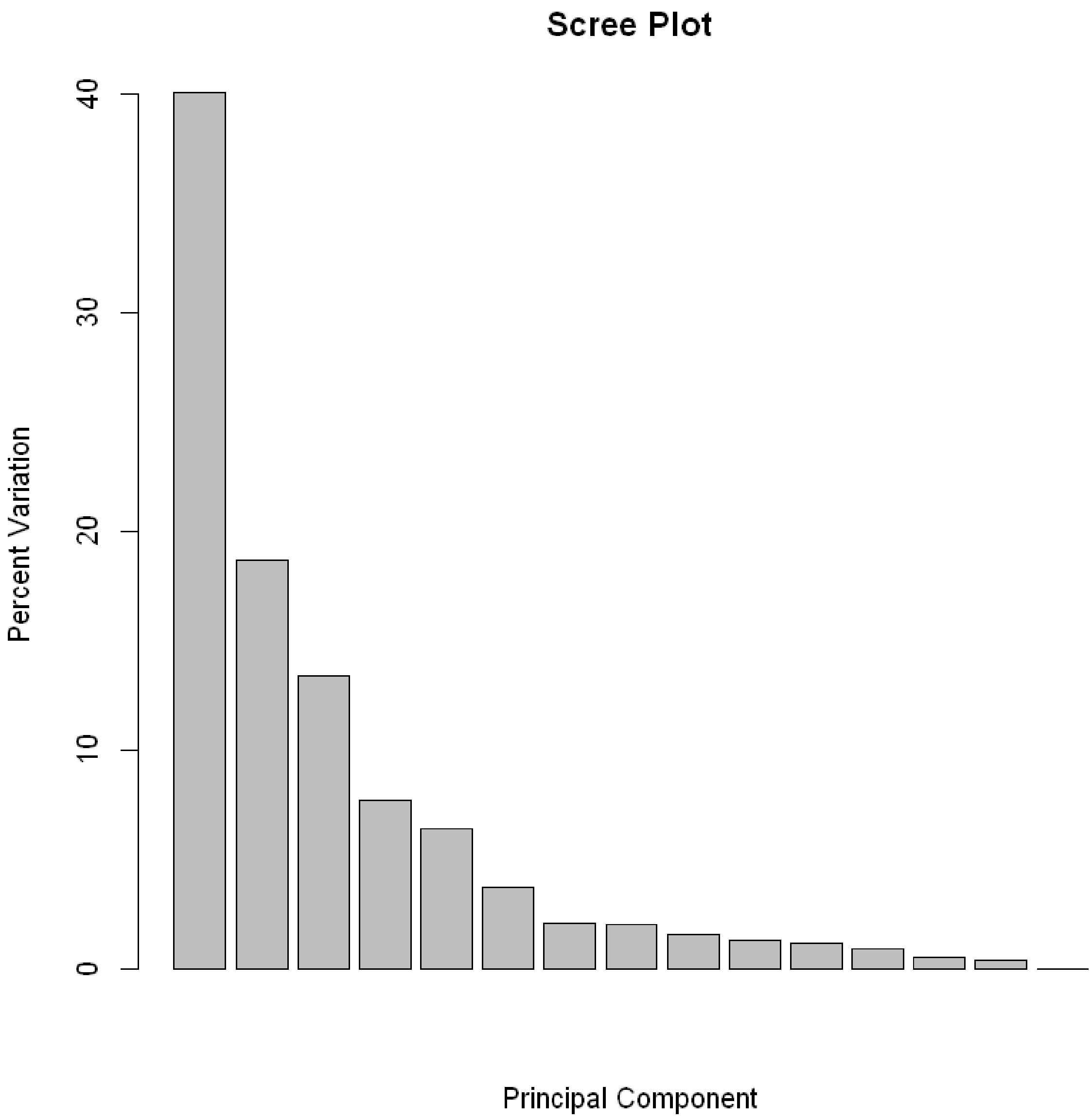
In [27]:
```
var.per
```

40.1   18.7   13.4   7.7   6.4   3.7   2.1   2   1.6   1.3   1.2   0.9   0.5   0.4   0

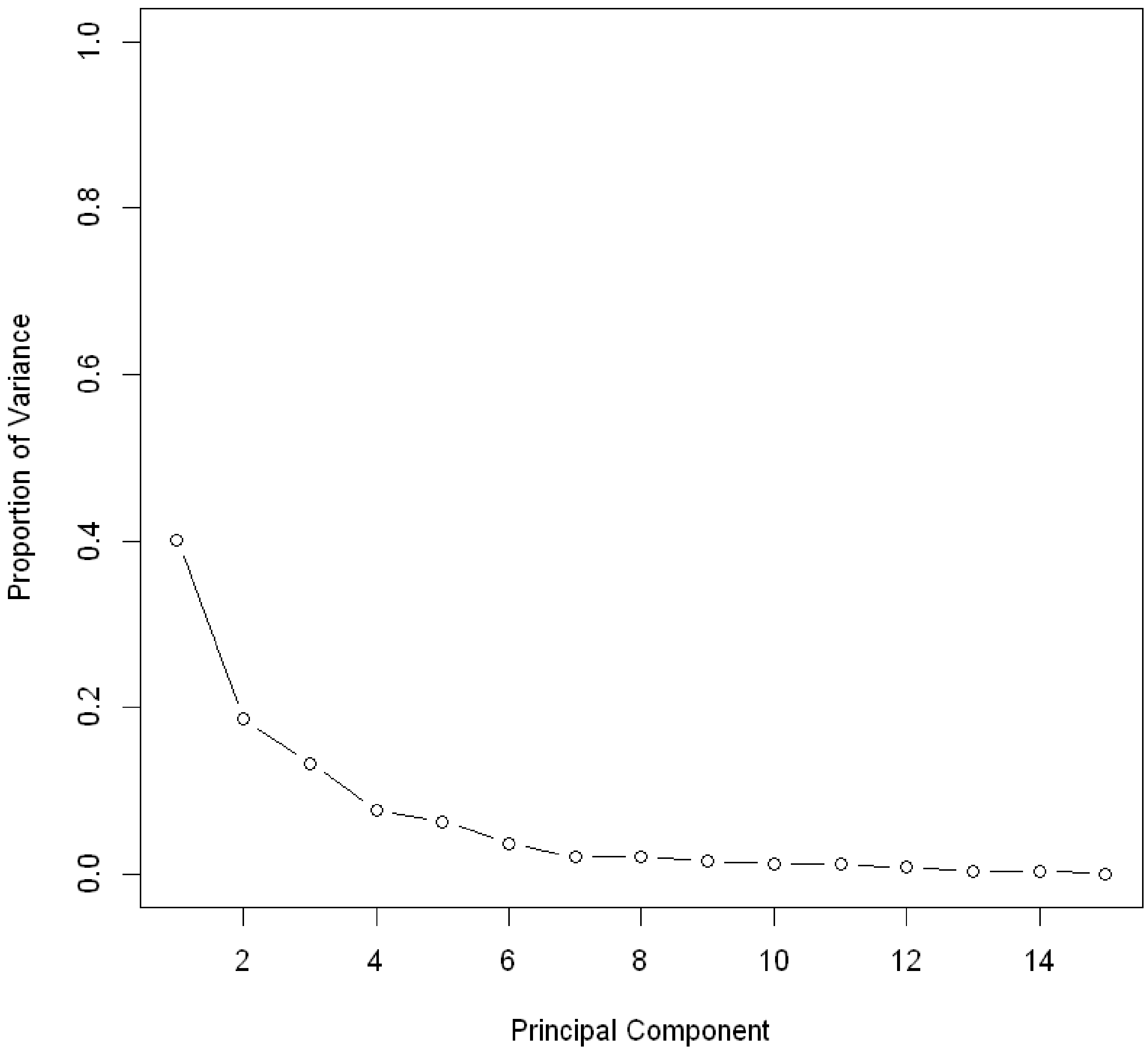
**We can see that the first principal component explains 40.1% variance. Second component explains 18.7% variance. Third component explains 13.4% variance and so on.**

# Selecting number of PCs' according to the following graphs

In [36]:
```
barplot(var.per, main="Scree Plot", xlab="Principal Component", ylab="Percent Variation")
```

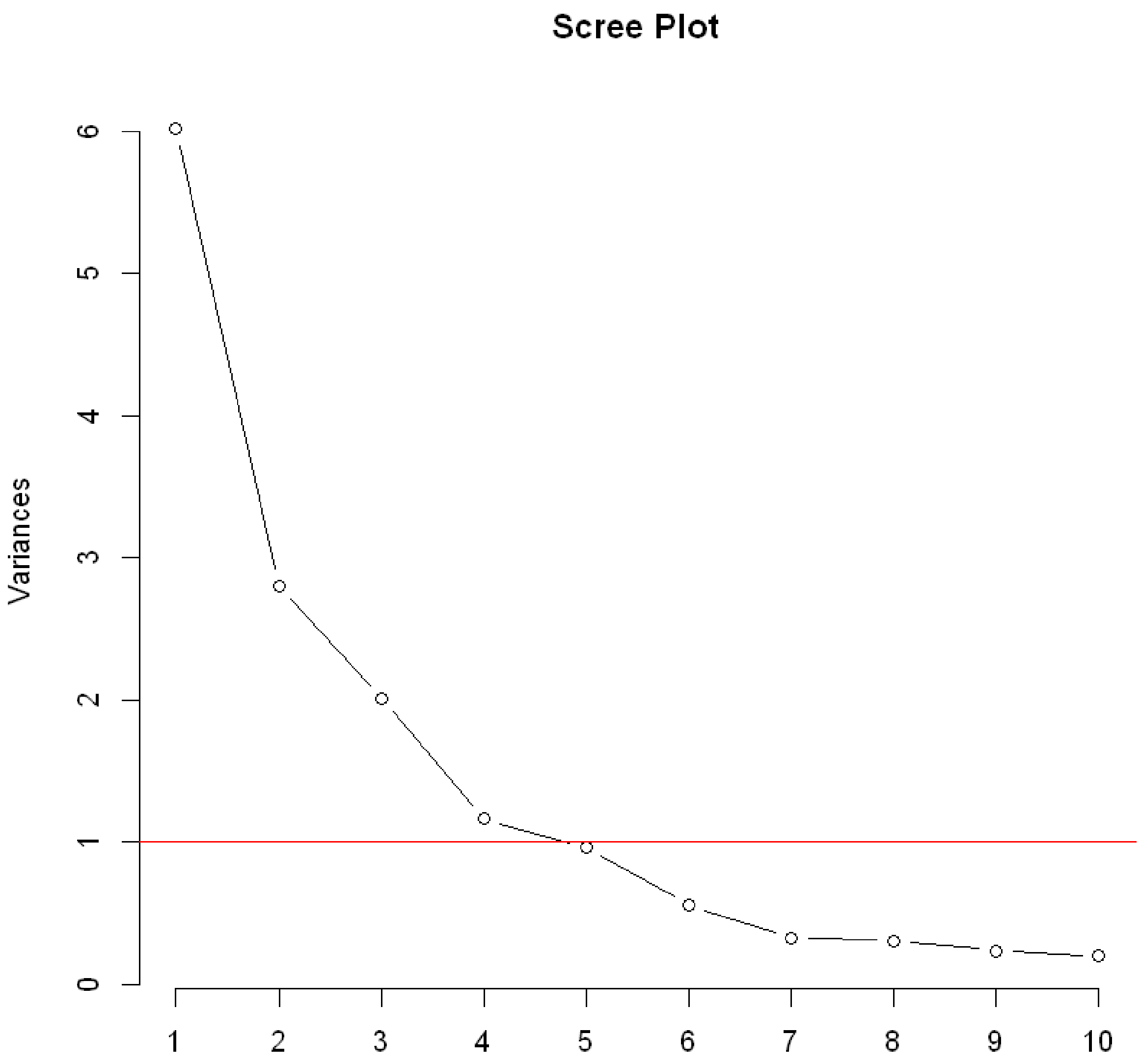


In [38]:
```
#The summary ranks the proportion of variance of each principal component. #PCs 1-3 have a significan
#relative to the rest. Shown in the plot below.
#get back eigenvalues, square the standard deviation
var <- my.prc$sdev^2
#get proportional variance by dividing each eigenvalue by the sum of eigenvalues
propvar <- var/sum(var)
plot(propvar,
     xlab = "Principal Component",
     ylab = "Proportion of Variance",
     ylim = c(0,1) , type= "b")
```



In [31]:

```
#Determine which PC variables are import. Kaiser method suggests any stdev greater than
#one is important.
screeplot(my.prc,main = "Scree Plot", type = "line")
abline(h=1, col="red")
```




### From the above plots; I'm taking the first 5 Principal Components, because they account for a little more than 85% of the variation in data

**based on this technique, we would choose to use the first 5 PCs in our model**

In [47]:
```
#we now combine PCs 1:k with the crime data from our original data set
PCcrime <- as.data.frame(cbind(my.prc$x[,1:5],dat[,16]))

colnames(PCcrime) <- c("PC1", "PC2", "PC3", "PC4", "PC5","Crime")
PCcrime
```

| PC1 | PC2 | PC3 | PC4 | PC5 | Crime |
|---|---|---|---|---|---|
| -4.1992835 | -1.09383120 | -1.11907395 | 0.67178115 | 0.055283376 | 791 |
| 1.1726630 | 0.67701360 | -0.05244634 | -0.08350709 | -1.173199821 | 1635 |
| -4.1737248 | 0.27677501 | -0.37107658 | 0.37793995 | 0.541345246 | 578 |
| 3.8349617 | -2.57690596 | 0.22793998 | 0.38262331 | -1.644746496 | 1969 |
| 1.8392999 | 1.33098564 | 1.27882805 | 0.71814305 | 0.041590320 | 1234 |
| 2.9072336 | -0.33054213 | 0.53288181 | 1.22140635 | 1.374360960 | 682 |
| 0.2457752 | -0.07362562 | -0.90742064 | 1.13685873 | 0.718644387 | 963 |
| -0.1301330 | -1.35985577 | 0.59753132 | 1.44045387 | -0.222781388 | 1555 |
| -3.6103169 | -0.68621008 | 1.28372246 | 0.55171150 | -0.324292990 | 856 |
| 1.1672376 | 3.03207033 | 0.37984502 | -0.28887026 | -0.646056610 | 705 |
| 2.5384879 | -2.66771358 | 1.54424656 | -0.87671210 | -0.324083561 | 1674 |
| 1.0065920 | -0.06044849 | 1.18861346 | -1.31261964 | 0.358087724 | 849 |
| 0.5161143 | 0.97485189 | 1.83351610 | -1.59117618 | 0.599881946 | 511 |
| 0.4265556 | 1.85044812 | 1.02893477 | -0.07789173 | 0.741887592 | 664 |
| -3.3435299 | 0.05182823 | -1.01358113 | 0.08840211 | 0.002969448 | 798 |
| -3.0310689 | -2.10295524 | -1.82993161 | 0.52347187 | -0.387454246 | 946 |
| -0.2262961 | 1.44939774 | -1.37565975 | 0.28960865 | 1.337784608 | 539 |
| -0.1127499 | -0.39407030 | -0.38836278 | 3.97985093 | 0.410914404 | 929 |
| 2.9195668 | -1.58646124 | 0.97612613 | 0.78629766 | 1.356288600 | 750 |
| 2.2998485 | -1.73396487 | -2.82423222 | -0.23281758 | -0.653038858 | 1225 |
| 1.1501667 | 0.13531015 | 0.28506743 | -2.19770548 | 0.084621572 | 742 |
| -5.6594827 | -1.09730404 | 0.10043541 | -0.05245484 | -0.689327990 | 439 |
| -0.1011749 | -0.57911362 | 0.71128354 | -0.44394773 | 0.689939865 | 1216 |
| 1.3836281 | 1.95052341 | -2.98485490 | -0.35942784 | -0.744371276 | 968 |
| 0.2727756 | 2.63013778 | 1.83189535 | 0.05207518 | 0.803692524 | 523 |
| 4.0565577 | 1.17534729 | -0.81690756 | 1.66990720 | -2.895110075 | 1993 |
| 0.8929694 | 0.79236692 | 1.26822542 | -0.57575615 | 1.830793964 | 342 |
| 0.1514495 | 1.44873320 | 0.10857670 | -0.51040146 | -1.023229895 | 1216 |
| 3.5592481 | -4.76202163 | 0.75080576 | 0.64692974 | 0.309946510 | 1043 |
| -4.1184576 | -0.38073981 | 1.43463965 | 0.63330834 | -0.254715638 | 696 |
| -0.6811731 | 1.66926027 | -2.88645794 | -1.30977099 | -0.470913997 | 373 |
| 1.7157269 | -1.30836339 | -0.55971313 | -0.70557980 | 0.331277622 | 754 |
| -1.8860627 | 0.59058174 | 1.43570145 | 0.18239089 | 0.291863659 | 1072 |
| 1.9526349 | 0.52395429 | -0.75642216 | 0.44289927 | 0.723474420 | 923 |
| 1.5888864 | -3.12998571 | -1.73107199 | -1.68604766 | 0.665406182 | 653 |
| 1.0709414 | -1.65628271 | 0.79436888 | -1.85172698 | 0.020031154 | 1272 |
| -4.1101715 | 0.15766712 | 2.36296974 | -0.56868399 | -2.469679496 | 831 |
| -0.7254706 | 2.89263339 | -0.36348376 | -0.50612576 | 0.028157162 | 566 |
| -3.3451254 | -0.95045293 | 0.19551398 | -0.27716645 | 0.487259213 | 826 |
| -1.0644466 | -1.05265304 | 0.82886286 | -0.12042931 | -0.645884788 | 1151 |
| 1.4933989 | 1.86712106 | 1.81853582 | -1.06112429 | 0.009855774 | 880 |
| -0.6789284 | 1.83156328 | -1.65435992 | 0.95121379 | 2.115630145 | 542 |
| -2.4164258 | -0.46701087 | 1.42808323 | 0.41149015 | -0.867397522 | 823 |
| 2.2978729 | 0.41865689 | -0.64422929 | -0.63462770 | -0.703116983 | 1030 |

| PC1 | PC2 | PC3 | PC4 | PC5 | Crime |
|---|---|---|---|---|---|
| -2.9245282 | -1.19488555 | -3.35139309 | -1.48966984 | 0.806659622 | 455 |
| 1.7654525 | 0.95655926 | 0.98576138 | 1.05683769 | 0.542466034 | 508 |
| 2.3125056 | 2.56161119 | -1.58223354 | 0.59863946 | -1.140712406 | 849 |

In [48]:
```
#using PCs combined with crime data, we create a linear regression model
#The advantage of doing this is to reduce the complexity of the model
#while also making it more robust
model <- lm(Crime~., data = PCcrime)
summary(model)
```

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

Residuals:
    Min      1Q  Median      3Q     Max 
-420.79 -185.01   12.21  146.24  447.86 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   905.09      35.59  25.428  < 2e-16 ***
PC1            65.22      14.67   4.447 6.51e-05 ***
PC2           -70.08      21.49  -3.261  0.00224 ** 
PC3            25.19      25.41   0.992  0.32725    
PC4            69.45      33.37   2.081  0.04374 *  
PC5          -229.04      36.75  -6.232 2.02e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 244 on 41 degrees of freedom
Multiple R-squared:  0.6452,	Adjusted R-squared:  0.6019 
F-statistic: 14.91 on 5 and 41 DF,  p-value: 2.446e-08
```

In [49]:
```
#now to do our transformation, we first need our intercept
beta0 <- model$coefficients[1]
beta0
```

**(Intercept):** 905.085106382979

In [51]:
```
#below we pull out our model coefficients, and make the Beta vector
betas <- model$coefficients[2:6]
betas
```

| | |
|---|---|
| **PC1** | 65.215930138666 |
| **PC2** | -70.0831185497858 |
| **PC3** | 25.1940780425771 |
| **PC4** | 69.4460307968377 |
| **PC5** | -229.042822001687 |

**Bringing back the model output to orginal variables**

In [94]:
```
#now multply the coefficients by our rotated matrix, A to create alpha vector
alpha <- my.prc$rotation[,1:5] %*% betas
t(alpha)
```

| M | So | Ed | Po1 | Po2 | LF | M.F | Pop | NW | U1 | U2 | Wealth | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 60.79435 | 37.84824 | 19.94776 | 117.3449 | 111.4508 | 76.2549 | 108.1266 | 58.88024 | 98.07179 | 2.866783 | 32.34551 | 35.93336 | 22.1 |

## BUT... these coefficients above are using scaled data.

**Now, we have to convert back to the original data.**

**When scaling, this function subtracts the mean and divides by the standard deviation, for each variable.**

**So, alpha * (x - mean)/sd = originalAlpha * x.**

Meaning:

**(1) originalAlpha = alpha/sd**

**(2) we have to modify the constant term a0 by alpha*mean/sd**

In [96]:
```
#we recover our original alpha values by dividing the alpha vector by sigma
#and our original beta by subtracting from the intercept the sum of (alpha*mu)/sigma
mu <- sapply(dat[,1:15],mean)
# print(paste("Mu:", mu))
sigma <- sapply(dat[,1:15],sd)
# print(paste("Sigma:", sigma))

origAlpha <- alpha/sigma
t(origAlpha)

origBeta0 <- beta0 - sum(alpha*mu /sigma)
print(paste("origBeta0:", origBeta0))
```

| M | So | Ed | Po1 | Po2 | LF | M.F | Pop | NW | U1 | U2 | Wealth | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 48.37374 | 79.01922 | 17.8312 | 39.48484 | 39.85892 | 1886.946 | 36.69366 | 1.546583 | 9.537384 | 159.0115 | 38.29933 | 0.03724014 | 5. |

```
[1] "origBeta0: -5933.83744880081"
```

In [62]:
```
#estimates now gives us our model Y = aX + b
#where a is our scaled alpha and b is our original intercept
estimates <- as.matrix(dat[,1:15]) %*% origAlpha + origBeta0
```

In [83]:
```
estimates
```

| |
|---|
| 713.6803 |
| 1195.7066 |
| 506.4008 |
| 1744.8151 |
| 1004.3223 |
| 901.3083 |
| 817.7618 |
| 1158.0158 |
| 862.6600 |
| 906.1942 |
| 1309.8473 |
| 831.7397 |
| 668.7175 |
| 653.8079 |
| 663.3242 |
| 933.7860 |
| 467.7924 |
| 1097.8331 |
| 975.2212 |
| 1238.8452 |
| 805.7895 |
| 769.6724 |
| 768.1369 |
| 928.9523 |
| 604.2355 |
| 1845.7567 |
| 480.4270 |
| 1015.0839 |
| 1463.7936 |
| 801.6455 |
| 687.8542 |
| 969.6941 |
| 722.6822 |
| 841.7013 |
| 914.9564 |
| 977.8353 |
| 1211.6890 |
| 604.2928 |
| 627.6148 |
| 1069.8938 |
| 841.4929 |
| 272.2545 |
| 1043.4520 |
| 1126.3430 |
| 425.4541 |
| 927.1627 |
| 1139.3538 |

In [85]:
```
#we can now use our estimates to calculate the R-squared values
#to observe the accuracy of our model
SSE = sum((estimates - dat[,16])^2)
print(paste("SSE:",SSE))
```

```
[1] "SSE: 2441394.04997516"
```

In [86]:
```
SStot = sum((dat[,16] - mean(dat[,16]))^2)
print(paste("SStot:",SStot))
```

```
[1] "SStot: 6880927.65957447"
```

In [87]:
```
R2 <- 1 - SSE/SStot
print(paste("R2:",R2))
```

```
[1] "R2: 0.645194053656692"
```

In [88]:
```
R2_adjust <- R2 - (1-R2)*5/(nrow(dat)-4)
print(paste("R2_adjust:",R2_adjust))
```

```
[1] "R2_adjust: 0.603937548267935"
```

In [89]:
```
#now we will use the new_city data given from last week to see what
#our improved model predicts the crime rate to be
new_city <- data.frame(M= 14.0, So = 0, Ed = 10.0, Po1 = 12.0, Po2 = 15.5,
                    LF = 0.640, M.F = 94.0, Pop = 150, NW = 1.1, U1 = 0.120, U2 = 3.6, Wealth = 3200,
                       Ineq = 20.1, Prob = 0.040,Time = 39.0)
```

In [90]:
```
#first we apply the PCA data onto the new city data so we can apply our model
pred_df <- data.frame(predict(my.prc, new_city))
```

In [91]:
```
pred_df
```

| PC1 | PC2 | PC3 | PC4 | PC5 | PC6 | PC7 | PC8 | PC9 | PC10 | PC11 | PC |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1.224044 | -2.767641 | 0.533605 | -1.146837 | -1.206098 | 2.333343 | -0.1535916 | -1.391625 | 1.460274 | -0.4525158 | -0.3466498 | 1.663 |

In [76]:
```
#now predict the Crime rate using Principal components and new city data
pred <- predict(model, pred_df)
```

In [77]:
```
pred
```

**1:** 1388.92569475604

**This value makes sense relative to the other Crime values**

**Relative to last weeks prediction of 155 and an R-squared of 0.8031, this model seems slightly less sufficient at prescribing values. But this was only a small test to compare, and we observed that with significantly less predictors, a PCA model can deliver nearly the same accuracy.**

## Question 10.1

***Using the same crime data set uscrime.txt as in Questions 8.2 and 9.1, find the best model you can using***

***(a) a regression tree model, and***

***(b) a random forest model.***

***In R, you can use the tree package or the rpart package, and the randomForest package. For each model, describe one or two qualitative takeaways you get from analyzing the results (i.e., don't just stop when you have a good model, but interpret it too).***

In [98]:
```
data <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)
```

In [100]:
```
head(data)
```

| M | So | Ed | Po1 | Po2 | LF | M.F | Pop | NW | U1 | U2 | Wealth | Ineq | Prob | Time | Crime |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 791 |
| 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 | 1635 |
| 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 | 578 |
| 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 | 1969 |
| 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 | 1234 |
| 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 | 682 |

# Using Regression Tree Model

In [102]:
```
########## Without splitting data into training and testing sets ##########

# install.packages("tree")
library(tree)

set.seed(1)
```

In [180]:
```
# Fit a regression tree function to the crime data

tree.data <- tree(Crime~., data = data)
summary(tree.data)
```

```
Error in eval(predvars, data, env): object 'Crime' not found
Traceback:

1. tree(Crime ~ ., data = data)
2. eval.parent(m)
3. eval(expr, p)
4. eval(expr, p)
5. model.frame.default(formula = Crime ~ ., data = data)
6. eval(predvars, data, env)
7. eval(predvars, data, env)
```

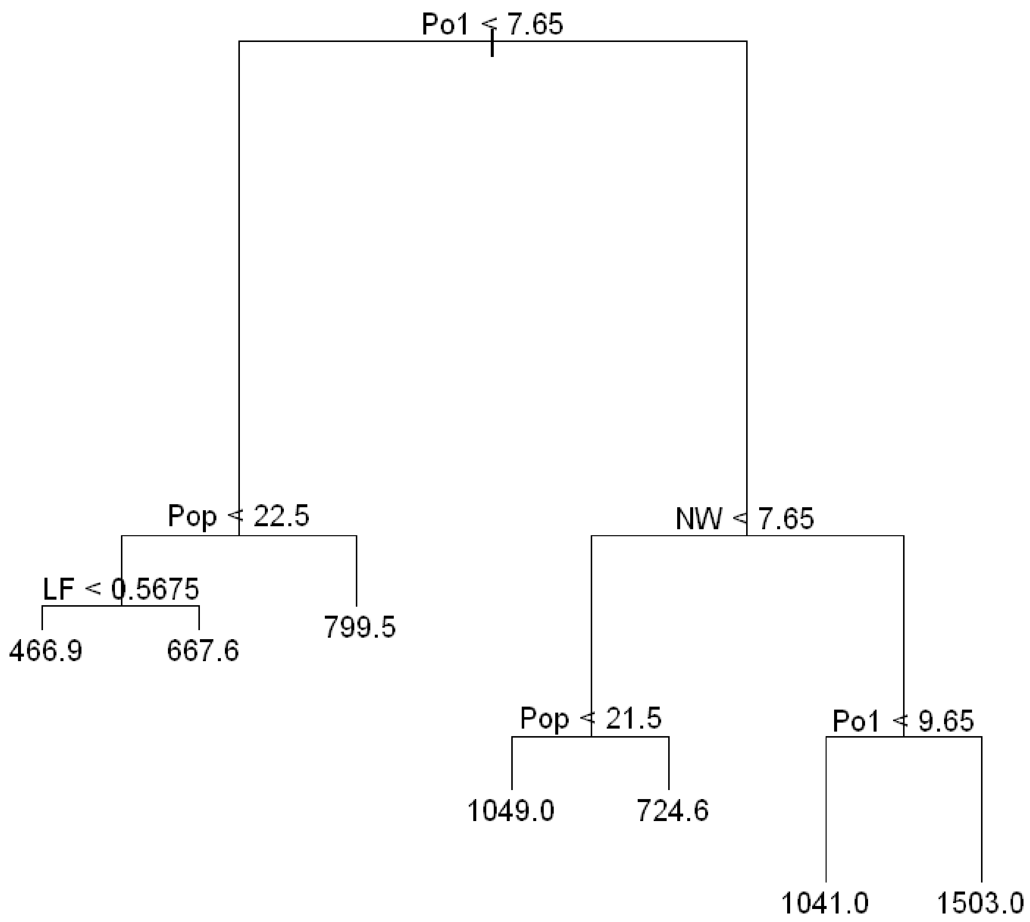
***Notice that only 4 predictors were used in the construction of this tree***

***More information about the way the tree was split***

In [181]:
```
print(tree.data$frame)
```

```
      var  n        dev      yval splits.cutleft splits.cutright
1     Po1 47 6880927.66  905.0851          <7.65           >7.65
2     Pop 23  779243.48  669.6087          <22.5           >22.5
4      LF 12  243811.00  550.5000        <0.5675         >0.5675
8  <leaf>  7   48518.86  466.8571
9  <leaf>  5   77757.20  667.6000
5  <leaf> 11  179470.73  799.5455
3      NW 24 3604162.50 1130.7500          <7.65           >7.65
6     Pop 10  557574.90  886.9000          <21.5           >21.5
12 <leaf>  5  146390.80 1049.2000
13 <leaf>  5  147771.20  724.6000
7     Po1 14 2027224.93 1304.9286          <9.65           >9.65
14 <leaf>  6  170828.00 1041.0000
15 <leaf>  8 1124984.88 1502.8750
```

In [107]:
```
# Plot the regression tree

plot(tree.data)

text(tree.data)
```




### From the graph:

***We can notice that Pop is used in 2 places. Also, the rightmost brach of the tree, it is also present 2 times once at top and then again at the bottom.***

***The model seems to be overfitted. We can either prune the data or use CV or PCA to handle it.***

***The model shows that Po1 is main factor for branching.***

In [128]:
```
# Calculate SSres of the unpruned regression model.

yhat <- predict(tree.data)
SSres <- sum((yhat-data$Crime)^2)
```

In [129]:
```
SStot <- sum((data$Crime - mean(data$Crime))^2)
R2 <- 1 - SSres/SStot
R2
```

0.724496208475934

The R^2 looks fine for Trees, Next lets look at **Random Forest**

# Using Random Forest Model

In [136]:
```
data <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)
```

In [137]:
```
# install.packages("randomForest")
library(randomForest)
set.seed(1)
```

In [138]:
```
# Grow the random tree and set the number of predictors that want to consider at each split of the tr

numpred <- 4 # How many variables to split or branch
rf.data <- randomForest(Crime~., data = data, mtry = numpred, importance = TRUE)
rf.data
```

```
Call:
 randomForest(formula = Crime ~ ., data = data, mtry = numpred,      importance = TRUE)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 4

          Mean of squared residuals: 82393.69
                    % Var explained: 43.72
```

In [139]:
```
# Calculate SSres of the random forest model

yhat.rf <- predict(rf.data)
SSres <- sum((yhat.rf-data$Crime)^2)
```

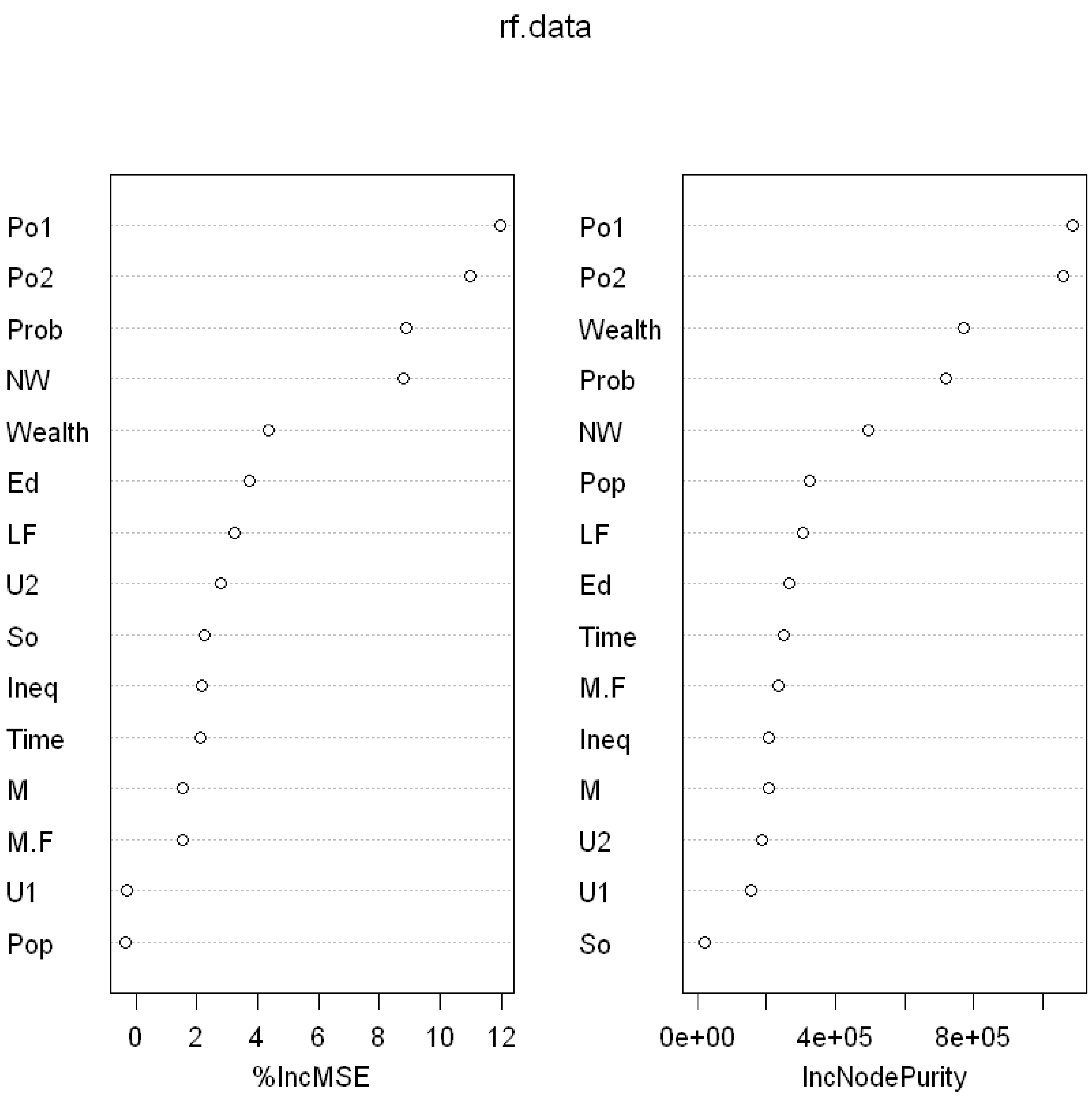
In [140]:
```
# Calculate SStot and R-squared of this model

SStot <- sum((data$Crime - mean(data$Crime))^2)
R2 <- 1 - SSres/SStot
R2
```

0.437211982904405

In [141]:
```
importance(rf.data)
```

| | %IncMSE | IncNodePurity |
|---|---|---|
| **M** | 1.5433378 | 205277.24 |
| **So** | 2.2457511 | 21269.39 |
| **Ed** | 3.7399142 | 264814.17 |
| **Po1** | 11.9531848 | 1084645.06 |
| **Po2** | 11.0005698 | 1057598.29 |
| **LF** | 3.2283145 | 304235.71 |
| **M.F** | 1.5315964 | 235479.84 |
| **Pop** | -0.3558573 | 325124.91 |
| **NW** | 8.7914688 | 495462.04 |
| **U1** | -0.3078521 | 155244.37 |
| **U2** | 2.7747464 | 187534.18 |
| **Wealth** | 4.3542465 | 770430.48 |
| **Ineq** | 2.1762932 | 206096.64 |
| **Prob** | 8.8829484 | 717873.42 |
| **Time** | 2.1279085 | 249099.59 |

In [143]:
```
# Plots of these importance measures

varImpPlot(rf.data)
```




**Po1 was the primary branching variable. So Both of the model thinks that Po1 is the important factor for crime. And, as we saw from regression trees, random forest gives better predictive quality for datapoints where Po1<7.65 than Po1>7.65. But, Random forst have predicted value for Po1>7.65**

### Question 10.2

***Describe a situation or problem from your job, everyday life, current events, etc., for which a logistic regression model would be appropriate. List some (up to 5) predictors that you might use.***

I have not worked on logistic regression. But as an avid soccer fan, I read multiple articles prior to the 2018 World cup how winner of each match and winner of the overall world cup could be predicted using regression modeling. The predictors used were:

1. average goals scored in 10 matches prior to the current match,
2. rank of each team playing in the match,
3. historic record of the team's result in that country,
4. historic average of number of fans of each team in the event location.
5. team strength

### Question 10.3(a)

***Using the GermanCredit data set germancredit.txt from [http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german (http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german)](http://archive.ics.uci.edu/ml/machine-learning-databases/statlog/german) / (description at [http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29 (http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29)](http://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29) ), use logistic regression to find a good predictive model for whether credit applicants are good credit risks or not. Show your model (factors used and their coefficients), the software output, and the quality of fit. You can use the glm function in R. To get a logistic regression (logit) model on data where the response is either zero or one, use family=binomial(link="logit") in your glm function call.***

In [145]:
```
data <- read.table("germancredit.txt",sep = " ")
```

In [147]:
```
head(data)
```

| V1 | V2 | V3 | V4 | V5 | V6 | V7 | V8 | V9 | V10 | ... | V12 | V13 | V14 | V15 | V16 | V17 | V18 | V19 | V20 | V21 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A11 | 6 | A34 | A43 | 1169 | A65 | A75 | 4 | A93 | A101 | ... | A121 | 67 | A143 | A152 | 2 | A173 | 1 | A192 | A201 | 1 |
| A12 | 48 | A32 | A43 | 5951 | A61 | A73 | 2 | A92 | A101 | ... | A121 | 22 | A143 | A152 | 1 | A173 | 1 | A191 | A201 | 2 |
| A14 | 12 | A34 | A46 | 2096 | A61 | A74 | 2 | A93 | A101 | ... | A121 | 49 | A143 | A152 | 1 | A172 | 2 | A191 | A201 | 1 |
| A11 | 42 | A32 | A42 | 7882 | A61 | A74 | 2 | A93 | A103 | ... | A122 | 45 | A143 | A153 | 1 | A173 | 2 | A191 | A201 | 1 |
| A11 | 24 | A33 | A40 | 4870 | A61 | A73 | 3 | A93 | A101 | ... | A124 | 53 | A143 | A153 | 2 | A173 | 2 | A191 | A201 | 2 |
| A14 | 36 | A32 | A46 | 9055 | A65 | A73 | 2 | A93 | A101 | ... | A124 | 35 | A143 | A153 | 1 | A172 | 2 | A192 | A201 | 1 |

In [148]:
```
str(data)
```

```
'data.frame':	1000 obs. of  21 variables:
 $ V1 : Factor w/ 4 levels "A11","A12","A13",..: 1 2 4 1 1 4 4 2 4 2 ...
 $ V2 : int  6 48 12 42 24 36 24 36 12 30 ...
 $ V3 : Factor w/ 5 levels "A30","A31","A32",..: 5 3 5 3 4 3 3 3 3 5 ...
 $ V4 : Factor w/ 10 levels "A40","A41","A410",..: 5 5 8 4 1 8 4 2 5 1 ...
 $ V5 : int  1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...
 $ V6 : Factor w/ 5 levels "A61","A62","A63",..: 5 1 1 1 1 5 3 1 4 1 ...
 $ V7 : Factor w/ 5 levels "A71","A72","A73",..: 5 3 4 4 3 3 5 3 4 1 ...
 $ V8 : int  4 2 2 2 3 2 3 2 2 4 ...
 $ V9 : Factor w/ 4 levels "A91","A92","A93",..: 3 2 3 3 3 3 3 3 1 4 ...
 $ V10: Factor w/ 3 levels "A101","A102",..: 1 1 1 3 1 1 1 1 1 1 ...
 $ V11: int  4 2 3 4 4 4 4 2 4 2 ...
 $ V12: Factor w/ 4 levels "A121","A122",..: 1 1 1 2 4 4 2 3 1 3 ...
 $ V13: int  67 22 49 45 53 35 53 35 61 28 ...
 $ V14: Factor w/ 3 levels "A141","A142",..: 3 3 3 3 3 3 3 3 3 3 ...
 $ V15: Factor w/ 3 levels "A151","A152",..: 2 2 2 3 3 3 2 1 2 2 ...
 $ V16: int  2 1 1 1 2 1 1 1 1 2 ...
 $ V17: Factor w/ 4 levels "A171","A172",..: 3 3 2 3 3 2 3 4 2 4 ...
 $ V18: int  1 1 2 2 2 2 1 1 1 1 ...
 $ V19: Factor w/ 2 levels "A191","A192": 2 1 1 1 1 2 1 2 1 1 ...
 $ V20: Factor w/ 2 levels "A201","A202": 1 1 1 1 1 1 1 1 1 1 ...
 $ V21: int  1 2 1 1 2 1 1 1 1 2 ...
```

In [149]:
```
# Since binomial family of glm recognises 0 and 1 as the classfication values,
# convert 1s and 2s to 0s and 1s for the response variable

data$V21[data$V21==1]<-0
data$V21[data$V21==2]<-1
```

In [150]:
```
# Set the seed to produce reproducible results as random sampling is done in the next step

set.seed(1)
```

In [182]:
```
# Divide the data into 70% training and 30% test/validation data

m <- nrow(data)
trn <- sample(1:m, size = round(m*0.7), replace = FALSE)
d.train <- data[trn,]
d.valid <- data[-trn,]
```

In [183]:
```
# Develop the logistic regression model

reg = glm(V21 ~.,family=binomial(link = "logit"),data=d.train)
summary(reg)
```

```
Call:
glm(formula = V21 ~ ., family = binomial(link = "logit"), data = d.train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.2662  -0.6349  -0.3339   0.6622   2.6297

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  1.0514129  1.2663260   0.830  0.40638
V1A12       -0.6652902  0.2733138  -2.434  0.01493 *
V1A13       -1.3939036  0.4711475  -2.959  0.00309 **
V1A14       -2.2252321  0.3053854  -7.287 3.18e-13 ***
V2           0.0253641  0.0111433   2.276  0.02284 *
V3A31       -0.1809034  0.6302029  -0.287  0.77407
V3A32       -0.9597747  0.4917581  -1.952  0.05097 .
V3A33       -0.8023087  0.5570455  -1.440  0.14978
V3A34       -1.0727642  0.4969643  -2.159  0.03088 *
V4A41       -1.9060301  0.4775558  -3.991 6.57e-05 ***
V4A410      -1.4298952  0.8610663  -1.661  0.09679 .
V4A42       -0.5778509  0.3259400  -1.773  0.07625 .
V4A43       -0.9798490  0.3094942  -3.166  0.00155 **
V4A44        0.1479986  0.9008533   0.164  0.86951
V4A45       -0.1138599  0.6634741  -0.172  0.86374
V4A46        0.5053416  0.4859994   1.040  0.29843
V4A48       -0.9245264  1.2441400  -0.743  0.45742
V4A49       -0.8934832  0.4174407  -2.140  0.03232 *
V5           0.0001691  0.0000525   3.221  0.00128 **
V6A62       -0.3328234  0.3676752  -0.905  0.36535
V6A63       -0.1007680  0.4431307  -0.227  0.82011
V6A64       -1.5759683  0.6370567  -2.474  0.01337 *
V6A65       -0.8211205  0.3118175  -2.633  0.00846 **
V7A72       -0.0341093  0.4916897  -0.069  0.94469
V7A73       -0.2543670  0.4700859  -0.541  0.58843
V7A74       -0.8333506  0.5154283  -1.617  0.10592
V7A75       -0.2373411  0.4745271  -0.500  0.61696
V8           0.4068087  0.1097667   3.706  0.00021 ***
V9A92       -0.1422847  0.4498370  -0.316  0.75177
V9A93       -1.0239869  0.4426357  -2.313  0.02070 *
V9A94       -0.0375507  0.5392126  -0.070  0.94448
V10A102     -0.1076950  0.5359027  -0.201  0.84073
V10A103     -1.5648611  0.6271165  -2.495  0.01258 *
V11         -0.0258543  0.1099363  -0.235  0.81407
V12A122      0.0691395  0.3162899   0.219  0.82697
V12A123     -0.0761277  0.2955972  -0.258  0.79676
V12A124      0.7693984  0.5234107   1.470  0.14157
V13         -0.0122375  0.0112988  -1.083  0.27878
V14A142     -0.3519180  0.5139900  -0.685  0.49355
V14A143     -0.8345949  0.3075896  -2.713  0.00666 **
V15A152      0.1843490  0.3018833   0.611  0.54142
V15A153     -0.3498265  0.5777246  -0.606  0.54483
V16         -0.0400713  0.2480178  -0.162  0.87165
V17A172      0.2356477  0.7372457   0.320  0.74925
V17A173      0.1320618  0.6990647   0.189  0.85016
V17A174      0.1260055  0.7052852   0.179  0.85821
V18          0.1883065  0.3021343   0.623  0.53312
V19A192      0.0180200  0.2460208   0.073  0.94161
V20A202     -1.1047091  0.7194261  -1.536  0.12465
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 851.79  on 699  degrees of freedom
Residual deviance: 597.52  on 651  degrees of freedom
AIC: 695.52

Number of Fisher Scoring iterations: 5
```

In [153]:
```
y_hat<-predict(reg,d.valid,type = "response")
y_hat
```

| | |
|---|---|
| **2** | 0.533555639960757 |
| **10** | 0.718510502211864 |
| **14** | 0.374509850783743 |
| **18** | 0.895250788614752 |
| **21** | 0.101522608296423 |
| **23** | 0.0632509239750618 |
| **24** | 0.0427049854444788 |
| **26** | 0.219686310909629 |
| **32** | 0.543029069758155 |
| **34** | 0.0693070078680152 |
| **38** | 0.223454437909501 |
| **46** | 0.231935574463956 |
| **47** | 0.10969106508214 |
| **50** | 0.137537806378329 |
| **53** | 0.0885670766835326 |
| **57** | 0.194246251321148 |

In [156]:
```
# y_hat is a vector of fractions.
# Now we can use a threshold to make yes/no decisions, and view the confusion matrix.

y_hat_round <- as.integer(y_hat > 0.5)
```

In [157]:
```
t <- table(y_hat_round,d.valid$V21)
t
```

```
y_hat_round   0   1
          0 183  48
          1  25  44
```

In [158]:
```
# Model's accuracy is (183 + 43) / (183 + 43 + 22 + 52) = 75%.

acc <- (t[1,1] + t[2,2]) / sum(t)
acc
```

0.756666666666667

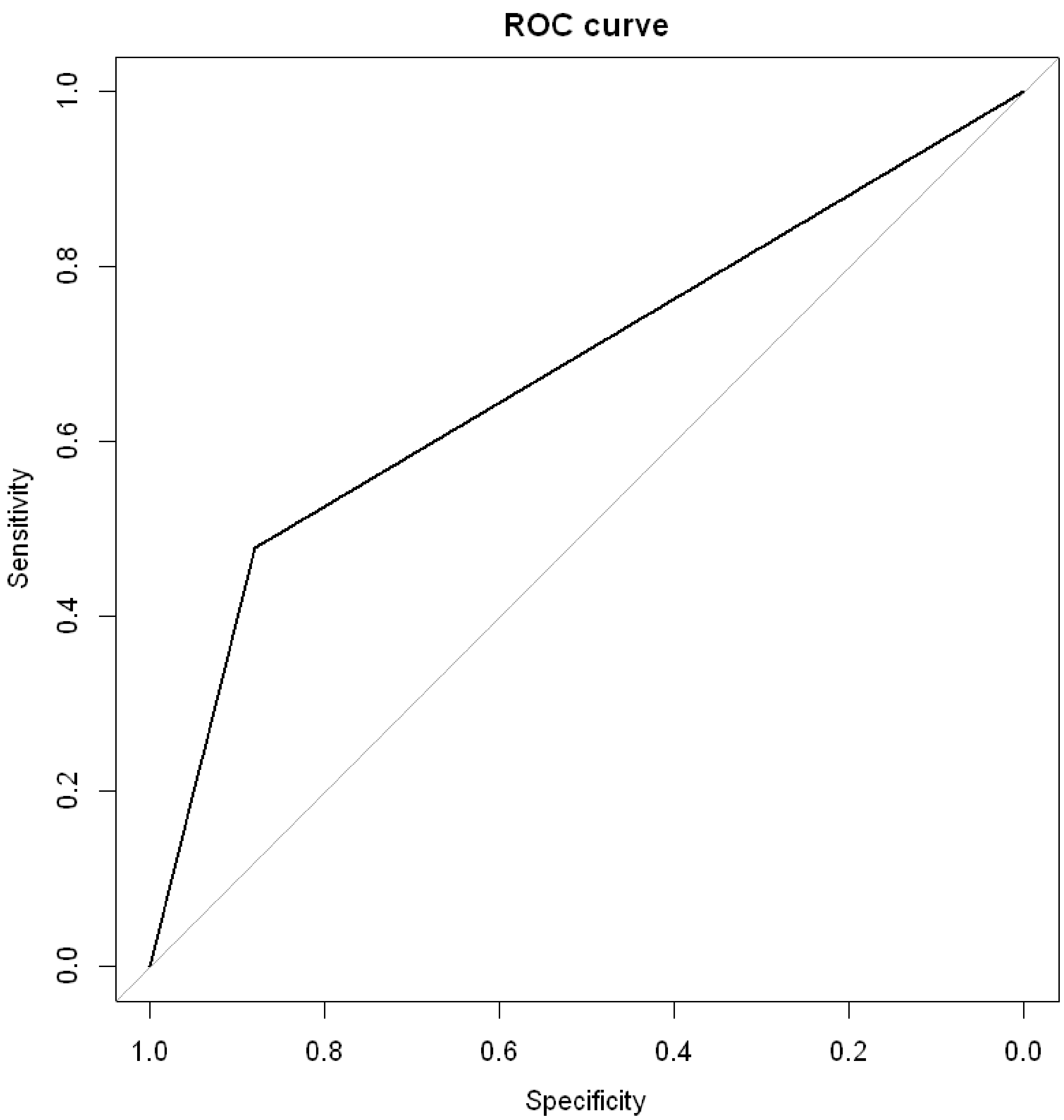
In [160]:
```
# Import the library for developing ROC curve

library(pROC)
```

In [161]:
```
# Develop ROC curve to determine the quality of fit

r<-roc(d.valid$V21,y_hat_round)
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

In [162]:
```
# Plot the ROC curve

plot(r,main="ROC curve")
r
```

```
Call:
roc.default(response = d.valid$V21, predictor = y_hat_round)

Data: y_hat_round in 208 controls (d.valid$V21 0) < 92 cases (d.valid$V21 1).
Area under the curve: 0.679
```



**AOC is 67.9% for a threshold of 50%, can AOC be better??? I guess I've to adjust the threshold.**

In [ ]:

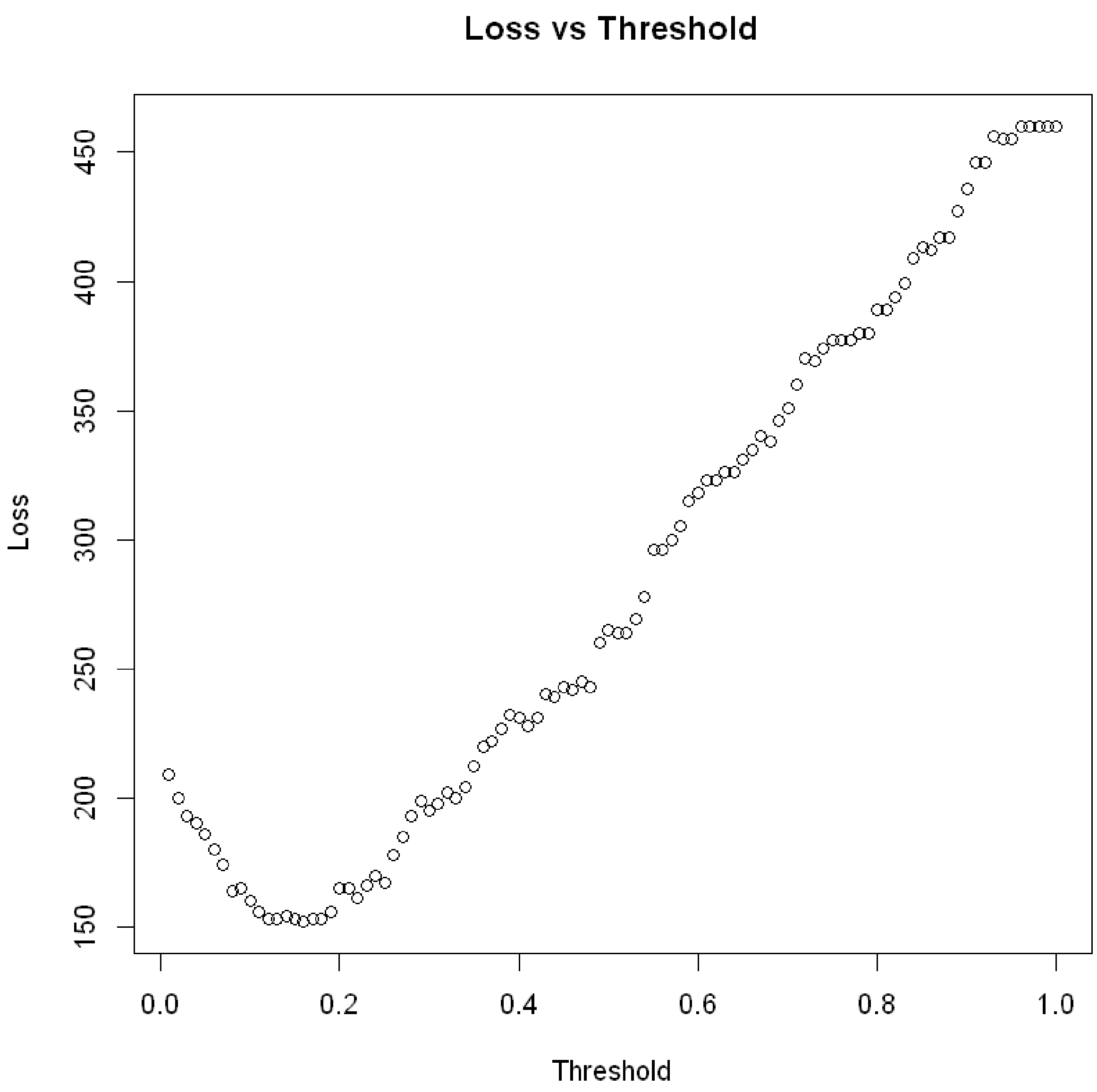
### Question 10.3(b)

**2. Because the model gives a result between 0 and 1, it requires setting a threshold probability to separate between "good" and "bad" answers. In this data set, they estimate that incorrectly identifying a bad customer as good, is 5 times worse than incorrectly classifying a good customer as bad. Determine a good threshold probability based on your model.**

In [164]:

```
# Writting a loop to calculate the loss for the value of thresholds ranging from 0.01 to 1.
# The loss of incorrectly classfying a "bad" customer is 5 times the loss of  incorrectly classifying

loss <- c()
for(i in 1:100)
{
  y_hat_round <- as.integer(y_hat > (i/100)) # This is to calculate threshold predictions

  tm <-as.matrix(table(y_hat_round,d.valid$V21))

  if(nrow(tm)>1) { c1 <- tm[2,1] } else { c1 <- 0 }
  if(ncol(tm)>1) { c2 <- tm[1,2] } else { c2 <- 0 }
  loss <- c(loss, c2*5 + c1)

}
```

In [165]:
```
plot(c(1:100)/100,loss,xlab = "Threshold",ylab = "Loss",main = "Loss vs Threshold")
```



In [167]:
```
loss
```

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 209 | 200 | 193 | 190 | 186 | 180 | 174 | 164 | 165 | 160 | 156 | 153 | 153 | 154 | 153 | 152 | 153 | 153 | 156 | 165 |
| 165 | 161 | 166 | 170 | 167 | 178 | 185 | 193 | 199 | 195 | 198 | 202 | 200 | 204 | 212 | 220 | 222 | 227 | 232 | 231 |
| 228 | 231 | 240 | 239 | 243 | 242 | 245 | 243 | 260 | 265 | 264 | 264 | 269 | 278 | 296 | 296 | 300 | 305 | 315 | 318 |
| 323 | 323 | 326 | 326 | 331 | 335 | 340 | 338 | 346 | 351 | 360 | 370 | 369 | 374 | 377 | 377 | 377 | 380 | 380 | 389 |
| 389 | 394 | 399 | 409 | 413 | 412 | 417 | 417 | 427 | 436 | 446 | 446 | 456 | 455 | 455 | 460 | 460 | 460 | 460 | 460 |

In [166]:
```
which.min(loss)
```

16

**The threshold probability corresponding to minimum expected loss is 0.16.**

In [168]:
```
#Here's the accuracy and area-under-curve for the 0.13 threshold:

y_hat_round <- as.integer(y_hat > (which.min(loss)/100)) # find 0/1 predictions
t <- table(y_hat_round,d.valid$V21)
t
```

```
y_hat_round   0   1
          0 106  10
          1 102  82
```

In [169]:
```
acc <- (t[1,1] + t[2,2]) / sum(t)
```

In [170]:
```
acc
```

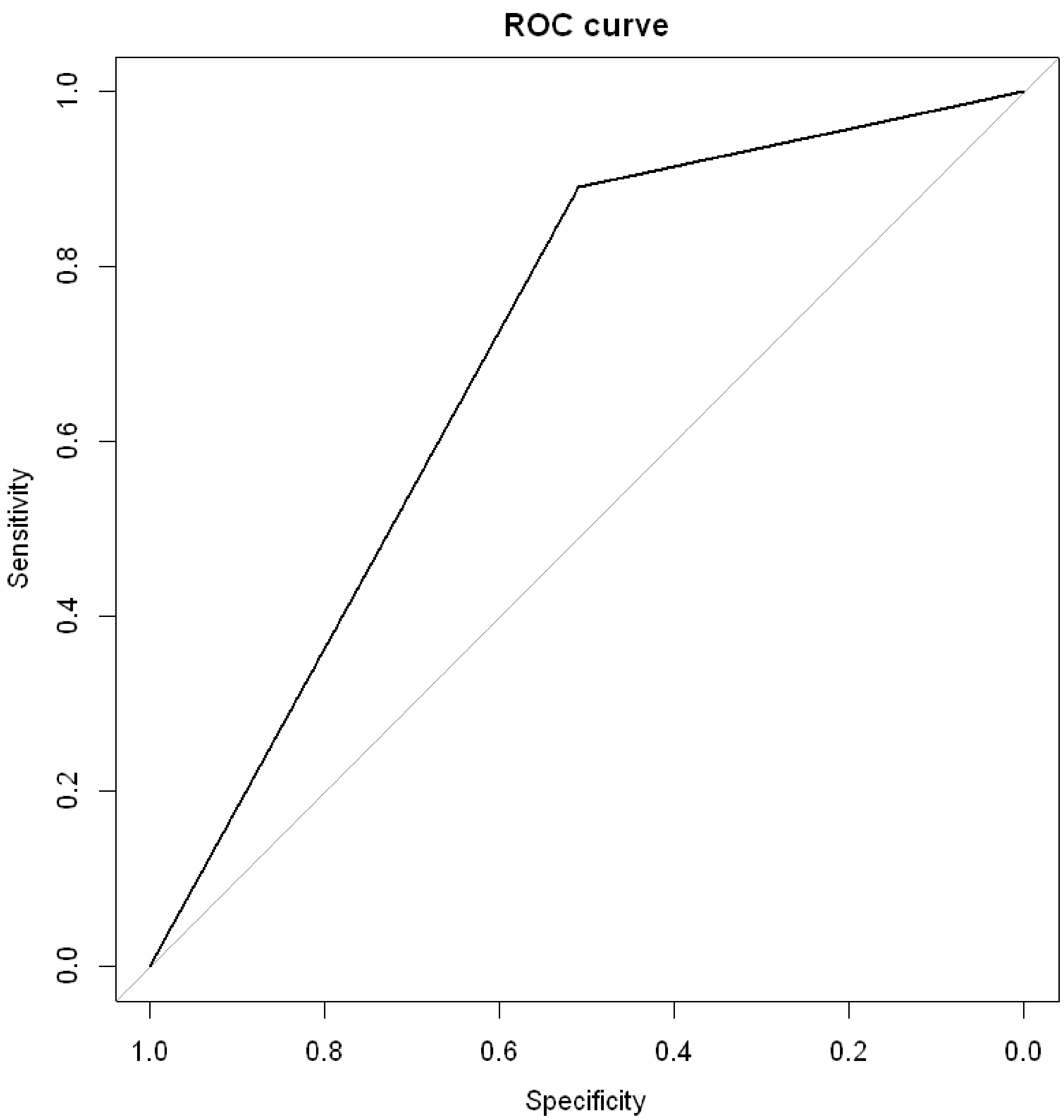
0.626666666666667

In [171]:
```
r<-roc(d.valid$V21,y_hat_round)
auc <- r$auc # get AUC
```

```
Setting levels: control = 0, case = 1
Setting direction: controls < cases
```

In [173]:
```
# PLot the ROC curve

plot(r,main="ROC curve")
r
```

```
Call:
roc.default(response = d.valid$V21, predictor = y_hat_round)

Data: y_hat_round in 208 controls (d.valid$V21 0) < 92 cases (d.valid$V21 1).
Area under the curve: 0.7005
```



***So keeping the threshold at 0.16 we are getting a AOC of 70%. Which is better than the result that we got before.***