ISYE6501 HW Wk4 Solution

```r
# First to set seed
set.seed(12)

# Load Package
library(ggplot2)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```r
library(tree)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

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)!)

First to check the function prcomp: Des: Performs a principal components analysis on the given data matrix and returns the results as an object of class prcomp.

```r
# Read us crime data file into R dataframe
df_crime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header=TRUE)
head(df_crime)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7
```

```
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6
##       Prob    Time Crime
## 1 0.084602 26.2011   791
## 2 0.029599 25.2999  1635
## 3 0.083401 24.3006   578
## 4 0.015801 29.9012  1969
## 5 0.041399 21.2998  1234
## 6 0.034201 20.9995   682
```

```r
str(df_crime)
```

```
## '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 ...
```

```r
summary(df_crime)
```

```
##        M               So               Ed             Po1       
##  Min.   :11.90   Min.   :0.0000   Min.   : 8.70   Min.   : 4.50  
##  1st Qu.:13.00   1st Qu.:0.0000   1st Qu.: 9.75   1st Qu.: 6.25  
##  Median :13.60   Median :0.0000   Median :10.80   Median : 7.80  
##  Mean   :13.86   Mean   :0.3404   Mean   :10.56   Mean   : 8.50  
##  3rd Qu.:14.60   3rd Qu.:1.0000   3rd Qu.:11.45   3rd Qu.:10.45  
##  Max.   :17.70   Max.   :1.0000   Max.   :12.20   Max.   :16.60  
##       Po2               LF              M.F              Pop        
##  Min.   : 4.100   Min.   :0.4800   Min.   : 93.40   Min.   :  3.00  
##  1st Qu.: 5.850   1st Qu.:0.5305   1st Qu.: 96.45   1st Qu.: 10.00  
##  Median : 7.300   Median :0.5600   Median : 97.70   Median : 25.00  
##  Mean   : 8.023   Mean   :0.5612   Mean   : 98.30   Mean   : 36.62  
##  3rd Qu.: 9.700   3rd Qu.:0.5930   3rd Qu.: 99.20   3rd Qu.: 41.50  
##  Max.   :15.700   Max.   :0.6410   Max.   :107.10   Max.   :168.00  
##        NW              U1                U2            Wealth    
##  Min.   : 0.20   Min.   :0.07000   Min.   :2.000   Min.   :2880  
##  1st Qu.: 2.40   1st Qu.:0.08050   1st Qu.:2.750   1st Qu.:4595  
##  Median : 7.60   Median :0.09200   Median :3.400   Median :5370  
##  Mean   :10.11   Mean   :0.09547   Mean   :3.398   Mean   :5254  
##  3rd Qu.:13.25   3rd Qu.:0.10400   3rd Qu.:3.850   3rd Qu.:5915  
##  Max.   :42.30   Max.   :0.14200   Max.   :5.800   Max.   :6890  
```

```
##       Ineq            Prob              Time            Crime
##  Min.   :12.60   Min.   :0.00690   Min.   :12.20   Min.   : 342.0
##  1st Qu.:16.55   1st Qu.:0.03270   1st Qu.:21.60   1st Qu.: 658.5
##  Median :17.60   Median :0.04210   Median :25.80   Median : 831.0
##  Mean   :19.40   Mean   :0.04709   Mean   :26.60   Mean   : 905.1
##  3rd Qu.:22.75   3rd Qu.:0.05445   3rd Qu.:30.45   3rd Qu.:1057.5
##  Max.   :27.60   Max.   :0.11980   Max.   :44.00   Max.   :1993.0
```

From above graph, we can see there is correlation between different variables. In order to improve the model, we need to use PCA to remove the correlation

As in 8.2, use the first 15 columns as data input

```r
# First list all the input variables
colnames(df_crime[,1:15])
```

```
##  [1] "M"      "So"     "Ed"     "Po1"    "Po2"    "LF"     "M.F"
##  [8] "Pop"    "NW"     "U1"     "U2"     "Wealth" "Ineq"   "Prob"
## [15] "Time"
```

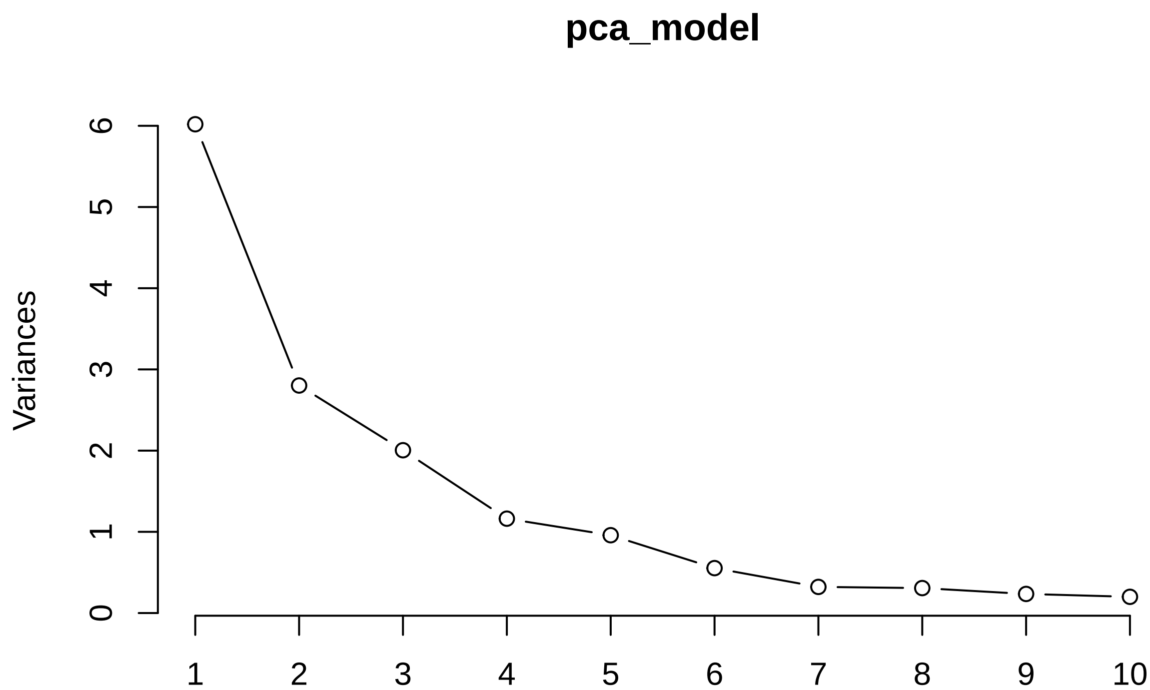
```r
# Build PCA Model, specifically to set the scale as TRUE
pca_model <- prcomp(df_crime[,1:15], scale. = TRUE)

# Summary of the model
summary(pca_model)
```

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

To visualize the result of PCA Model

```r
# use screeplot to plot the variances among variables
screeplot(pca_model, type='lines')
```



From above plot, we can see the first 3 PC have relatively big percentage of explained variances above 10%. WE can use the first 3 PC to build the regression model

```
# From above plot, we will only use the first 3 PCs for the regression.
# First to show the PC1-PC3 rotation of the variables
pca_rotation <- pca_model$rotation[,1:3]

# Use the scaled input data
pca_input <- pca_model$x[,1:3]

# Use the new PC1-PC3 and the reponse data to build the new dataframe to generate regression
pca_df <- as.data.frame(cbind(pca_input, df_crime[,16]))
colnames(pca_df)
```

```
## [1] "PC1" "PC2" "PC3" "V4"
```

From above, to make sure the new input data is created as dataframe (tried without as.data.frame, there is no column name for the response data)

```
pca_lm <- lm(V4~., data=pca_df)
summary(pca_lm)
```

```
##
## Call:
## lm(formula = V4 ~ ., data = pca_df)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -597.7 -225.4    6.8  153.2  926.3
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      49.80  18.175  < 2e-16 ***
## PC1            65.22      20.52   3.179  0.00274 **
## PC2           -70.08      30.07  -2.331  0.02454 *
## PC3            25.19      35.55   0.709  0.48233
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 341.4 on 43 degrees of freedom
## Multiple R-squared:  0.2716, Adjusted R-squared:  0.2208
## F-statistic: 5.346 on 3 and 43 DF,  p-value: 0.003217
```

For the model using PCA, the R-squared is only around 0.22. Comparing to the solution of 8.2 (The solution R square was about 0.72), the PCA linear regression seems that it does not have a result as good as linear regression.

```
# Get the coefficient for original variables

# First to get the Intercept
pca_lm$coefficients[1]
```

```
## (Intercept)
##    905.0851
```

```
# For the PC1 to PC3
pca_co <- pca_lm$coefficients[2:4]

# Original
ori <- pca_rotation %*% pca_co
t(ori)
```

```
##               M        So       Ed      Po1      Po2        LF       M.F
## [1,] -19.86436 -10.08792 8.814723 40.3135 40.11797 -4.056143 -25.16539
##            Pop        NW        U1       U2  Wealth      Ineq      Prob
## [1,] 42.06024 -1.181092 -14.52817 6.210256 30.4259 -21.93441 -30.94354
##           Time
## [1,] 30.92922
```

It is the scaled cofficient.

Not sure how to do the unscale to the original variables…

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 random Forest 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).

a Regression tree model

```
# Load raw data to R (Load the data again for this new question)
df_crime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)
```

use Tree function to build model

```
# Build model in Tree (CART)
tree_crime <- tree(Crime~., data = df_crime)
summary(tree_crime)
```

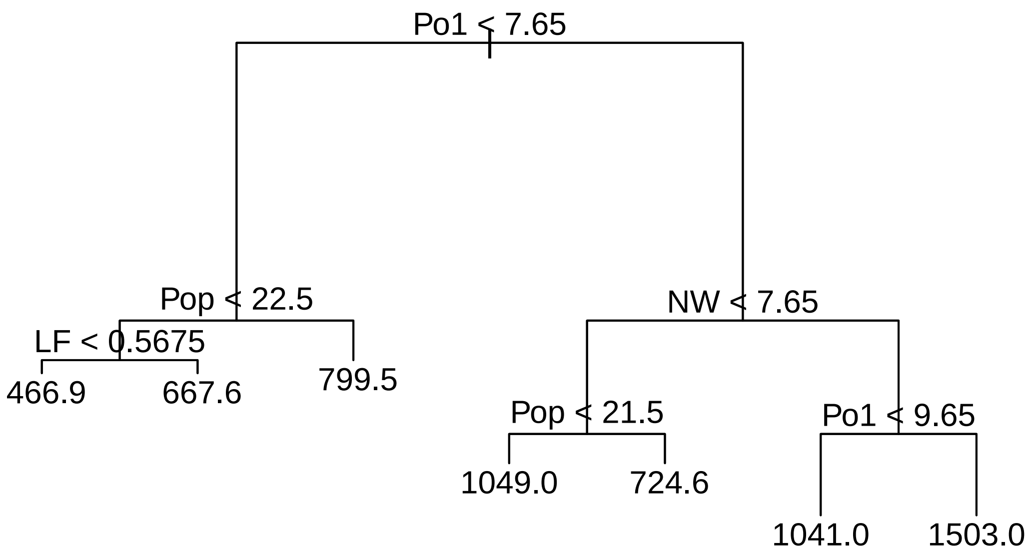
```
##
## Regression tree:
## tree(formula = Crime ~ ., data = df_crime)
## Variables actually used in tree construction:
## [1] "Po1" "Pop" "LF"  "NW"
## Number of terminal nodes:  7
## Residual mean deviance:  47390 = 1896000 / 40
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -573.900  -98.300   -1.545    0.000  110.600  490.100
```

From the above model summary, we can see all variables are actually used: Po1, Pop, LF and NW. The total nodes is 7.

```
# Show the frame of the tree model
tree_frame <- tree_crime$frame
tree_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
```

```
plot(tree_crime)
text(tree_crime)
```



The tree model as above. Not sure what would be the best way to prune the tree or improve the model.

b: Random Forest model

```
forest_model <- randomForest(Crime~., data = df_crime)
forest_model
```

```
##
## Call:
##  randomForest(formula = Crime ~ ., data = df_crime)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 5
##
##           Mean of squared residuals: 83361.58
##                     % Var explained: 43.06
```

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.

During the daily life, I need to make the decision if I should take the toll or not for daily commute. Preditor is usually as: 1. The time I leave home. Earlier time has light traffic so there will be no need for toll. 2. Weather. Bad weather will cause more traffic. 3. Season. Like during summer vacation or year end, the traffic is light so there is no need to take toll. 4. The number of daily meeting. If it is a busy day, might need to take toll to get to work faster.

Question 10.3

1. Using the GermanCredit data set germancredit.txt from 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), 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.

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.

1

```
# Load data
# Read the data file
df_ger<-read.table("germancredit.txt",sep = " ")
head(df_ger)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172
## 4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172
##   V18  V19  V20 V21
## 1   1 A192 A201   1
## 2   1 A191 A201   2
## 3   2 A191 A201   1
## 4   2 A191 A201   1
## 5   2 A191 A201   2
## 6   2 A192 A201   1
```

```
dim(df_ger)
```

```
## [1] 1000   21
```

From above, the V21 is the response data, the binary value needs to change from 1, 2 to 0, 1

```
# Update response value
df_ger$V21[df_ger$V21==1]<-0
df_ger$V21[df_ger$V21==2]<-1
head(df_ger)
```

```
##    V1 V2  V3  V4   V5  V6  V7 V8  V9  V10 V11  V12 V13  V14  V15 V16  V17
## 1 A11  6 A34 A43 1169 A65 A75  4 A93 A101   4 A121  67 A143 A152   2 A173
## 2 A12 48 A32 A43 5951 A61 A73  2 A92 A101   2 A121  22 A143 A152   1 A173
## 3 A14 12 A34 A46 2096 A61 A74  2 A93 A101   3 A121  49 A143 A152   1 A172
## 4 A11 42 A32 A42 7882 A61 A74  2 A93 A103   4 A122  45 A143 A153   1 A173
```

```
## 5 A11 24 A33 A40 4870 A61 A73  3 A93 A101   4 A124  53 A143 A153   2 A173
## 6 A14 36 A32 A46 9055 A65 A73  2 A93 A101   4 A124  35 A143 A153   1 A172
##   V18  V19  V20 V21
## 1   1 A192 A201   0
## 2   1 A191 A201   1
## 3   2 A191 A201   0
## 4   2 A191 A201   0
## 5   2 A191 A201   1
## 6   2 A192 A201   0
```

From the data, not sure in the model how glm will deal with those factor values.

```
# Build Logistic model
log_model <- glm(V21 ~.,data=df_ger,family=binomial(link = "logit"))
summary(log_model)
```

```
##
## Call:
## glm(formula = V21 ~ ., family = binomial(link = "logit"), data = df_ger)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.3410  -0.6994  -0.3752   0.7095   2.6116
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  4.005e-01  1.084e+00   0.369 0.711869
## V1A12       -3.749e-01  2.179e-01  -1.720 0.085400 .
## V1A13       -9.657e-01  3.692e-01  -2.616 0.008905 **
## V1A14       -1.712e+00  2.322e-01  -7.373 1.66e-13 ***
## V2           2.786e-02  9.296e-03   2.997 0.002724 **
## V3A31        1.434e-01  5.489e-01   0.261 0.793921
## V3A32       -5.861e-01  4.305e-01  -1.362 0.173348
## V3A33       -8.532e-01  4.717e-01  -1.809 0.070470 .
## V3A34       -1.436e+00  4.399e-01  -3.264 0.001099 **
## V4A41       -1.666e+00  3.743e-01  -4.452 8.51e-06 ***
## V4A410      -1.489e+00  7.764e-01  -1.918 0.055163 .
## V4A42       -7.916e-01  2.610e-01  -3.033 0.002421 **
## V4A43       -8.916e-01  2.471e-01  -3.609 0.000308 ***
## V4A44       -5.228e-01  7.623e-01  -0.686 0.492831
## V4A45       -2.164e-01  5.500e-01  -0.393 0.694000
## V4A46        3.628e-02  3.965e-01   0.092 0.927082
## V4A48       -2.059e+00  1.212e+00  -1.699 0.089297 .
## V4A49       -7.401e-01  3.339e-01  -2.216 0.026668 *
## V5           1.283e-04  4.444e-05   2.887 0.003894 **
## V6A62       -3.577e-01  2.861e-01  -1.250 0.211130
## V6A63       -3.761e-01  4.011e-01  -0.938 0.348476
## V6A64       -1.339e+00  5.249e-01  -2.551 0.010729 *
## V6A65       -9.467e-01  2.625e-01  -3.607 0.000310 ***
## V7A72       -6.691e-02  4.270e-01  -0.157 0.875475
## V7A73       -1.828e-01  4.105e-01  -0.445 0.656049
## V7A74       -8.310e-01  4.455e-01  -1.866 0.062110 .
## V7A75       -2.766e-01  4.134e-01  -0.669 0.503410
```

```
## V8           3.301e-01  8.828e-02   3.739 0.000185 ***
## V9A92       -2.755e-01  3.865e-01  -0.713 0.476040
## V9A93       -8.161e-01  3.799e-01  -2.148 0.031718 *
## V9A94       -3.671e-01  4.537e-01  -0.809 0.418448
## V10A102      4.360e-01  4.101e-01   1.063 0.287700
## V10A103     -9.786e-01  4.243e-01  -2.307 0.021072 *
## V11          4.776e-03  8.641e-02   0.055 0.955920
## V12A122      2.814e-01  2.534e-01   1.111 0.266630
## V12A123      1.945e-01  2.360e-01   0.824 0.409743
## V12A124      7.304e-01  4.245e-01   1.721 0.085308 .
## V13         -1.454e-02  9.222e-03  -1.576 0.114982
## V14A142     -1.232e-01  4.119e-01  -0.299 0.764878
## V14A143     -6.463e-01  2.391e-01  -2.703 0.006871 **
## V15A152     -4.436e-01  2.347e-01  -1.890 0.058715 .
## V15A153     -6.839e-01  4.770e-01  -1.434 0.151657
## V16          2.721e-01  1.895e-01   1.436 0.151109
## V17A172      5.361e-01  6.796e-01   0.789 0.430160
## V17A173      5.547e-01  6.549e-01   0.847 0.397015
## V17A174      4.795e-01  6.623e-01   0.724 0.469086
## V18          2.647e-01  2.492e-01   1.062 0.288249
## V19A192     -3.000e-01  2.013e-01  -1.491 0.136060
## V20A202     -1.392e+00  6.258e-01  -2.225 0.026095 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.73  on 999  degrees of freedom
## Residual deviance:  895.82  on 951  degrees of freedom
## AIC: 993.82
##
## Number of Fisher Scoring iterations: 5
```

The result is not really what I expected because the model created more variables beyond varialbe V1-V20. Like what we did in the linear regression model, instead of using all variables (might result to overfit), we can use the selected variables with more significance.

Selected variables: V1, V4, V6, V8

```
log_model_2 <- glm(V21 ~ V1+V4+V6+V8,,data=df_ger,family=binomial(link = "logit"))
summary(log_model_2)
```

```
##
## Call:
## glm(formula = V21 ~ V1 + V4 + V6 + V8, family = binomial(link = "logit"),
##     data = df_ger)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6158  -0.8240  -0.5018   1.0596   2.5546
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.15537    0.27470  -0.566 0.571668
```

```
## V1A12       -0.31208    0.18824  -1.658 0.097343 .
## V1A13       -1.14484    0.33723  -3.395 0.000687 ***
## V1A14       -1.83906    0.20846  -8.822  < 2e-16 ***
## V4A41       -0.96569    0.32054  -3.013 0.002589 **
## V4A410      -0.24209    0.61855  -0.391 0.695508
## V4A42       -0.36817    0.22759  -1.618 0.105720
## V4A43       -0.72016    0.21638  -3.328 0.000874 ***
## V4A44       -0.32315    0.69064  -0.468 0.639858
## V4A45       -0.16886    0.50477  -0.335 0.737980
## V4A46        0.39348    0.34805   1.131 0.258246
## V4A48       -1.63530    1.11754  -1.463 0.143387
## V4A49       -0.04479    0.27569  -0.162 0.870953
## V6A62       -0.05856    0.25129  -0.233 0.815739
## V6A63       -0.39712    0.36437  -1.090 0.275771
## V6A64       -1.25833    0.47108  -2.671 0.007559 **
## V6A65       -0.69720    0.23136  -3.013 0.002583 **
## V8           0.18776    0.07020   2.675 0.007483 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1221.7  on 999  degrees of freedom
## Residual deviance: 1041.4  on 982  degrees of freedom
## AIC: 1077.4
##
## Number of Fisher Scoring iterations: 5
```