Freddie Mac

Yazhe

9/29/2016

$min\frac{1}{2N}\sum_{i=1}^{N}(y_i-\beta_0-\beta x_i^{\mathrm{T}})^2+\lambda[(1-\alpha)*\|\beta\|_2^2/2+\alpha*\|\beta\|_1]$

```r
library(ROCR)
```

```
## Loading required package: gplots

##
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':
##
##     lowess
```

```r
library(Deducer)
```

```
## Loading required package: ggplot2

## Loading required package: JGR

## Loading required package: rJava

## Loading required package: JavaGD

## Loading required package: iplots

## Note: On Mac OS X we strongly recommend using iplots from within JGR.
## Proceed at your own risk as iplots cannot resolve potential ev.loop deadlocks.
## 'Yes' is assumed for all dialogs as they cannot be shown without a deadlock,
## also ievent.wait() is disabled.
## More recent OS X version do not allow signle-threaded GUIs and will fail.

##
## Please type JGR() to launch console. Platform specific launchers (.exe and .app) can also be obtaine

## Loading required package: car

## Warning: package 'car' was built under R version 3.2.5

## Loading required package: MASS

##
##
## Note Non-JGR console detected:
##  Deducer is best used from within JGR (http://jgr.markushelbig.org/).
##  To Bring up GUI dialogs, type deducer().
```

```r
library(grid)
library(devtools)
```

```
## Warning: package 'devtools' was built under R version 3.2.5
```

```r
library(car)
library(ggplot2)
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 3.2.5
```

```r
library(easyGgplot2)
library(glmnet)
```

```
## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.2.5

## Loading required package: foreach

## Loaded glmnet 2.0-5
```

```r
library(maps)
```

```
## Warning: package 'maps' was built under R version 3.2.5

##
## Attaching package: 'maps'

## The following object is masked from 'package:plyr':
##
##     ozone
```

```r
library(choroplethr)
```

```
## Loading required package: acs

## Loading required package: stringr

## Warning: package 'stringr' was built under R version 3.2.5

## Loading required package: XML

##
## Attaching package: 'acs'

## The following object is masked from 'package:base':
##
##     apply
```

```r
load("~/Desktop/Freddie_Mac_data/USMortgages2008_2009.rdata")
```

```r
names(D1)
```

```
##  [1] "seqno"               "score"               "first.pay.date"
##  [4] "first.time.homebuyer" "maturity.date"       "MSA"
##  [7] "insurance"           "number.units"        "occupancy.status"
## [10] "CLTV"                "DTI"                 "UPB"
## [13] "LTV"                 "OIR"                 "channel"
## [16] "PPM"                 "product.type"        "property.state"
## [19] "property.type"       "postal.code"         "loan.purpose"
## [22] "orig.loan.term"      "number.borrowers"    "seller"
## [25] "servicer"            "loan_age"            "def_flag"
```

simply remove NA first

Firstly, I delete column “seqno, first.pay.data, MSA, maturity.data, product.type, property.state, postal.code, loan_age”.

```r
newD = D1[,-c(1,3,6,5,17,18,20,26)]
na_count <- sapply(newD, function(x) sum(is.na(x))); na_count
```

```
##                score first.time.homebuyer            insurance
##                  843                    0                    6
##         number.units     occupancy.status                 CLTV
##                    1                    0                   81
##                  DTI                  UPB                  LTV
##                26985                    0                   81
##                  OIR              channel                  PPM
##                    0                    0                    0
##        property.type         loan.purpose       orig.loan.term
##                    0                    0                    0
##     number.borrowers               seller             servicer
##                  722                    0                    0
##             def_flag
##                    0
```

```r
D1_removeNA = newD[complete.cases(newD),]
rm(newD)
names(D1_removeNA)
```

```
##  [1] "score"               "first.time.homebuyer" "insurance"
##  [4] "number.units"        "occupancy.status"    "CLTV"
##  [7] "DTI"                 "UPB"                 "LTV"
## [10] "OIR"                 "channel"             "PPM"
## [13] "property.type"       "loan.purpose"        "orig.loan.term"
## [16] "number.borrowers"    "seller"              "servicer"
## [19] "def_flag"
```

we can see most deleted rows are caused by missing value from DTI, 26985 DTI are missed, more than 1% of the number of whole rows

categorical variables

first.time.homebuyer, insurance,number.units, occupance.status,channle, product.type, property.state, property.type, loan.purpose, orig.loan.term, seller,servicer, loan_age,PPM,

I also choose to translate "insurance" to categorical variables

```
D1_removeNA$insurance[which(D1_removeNA$insurance == 0)] = '0'
D1_removeNA$insurance[which(D1_removeNA$insurance != 0)] = '1'
```

```
factor_data = subset(D1_removeNA, select=c("first.time.homebuyer","insurance",
                                           "number.units","occupancy.status",
                                           "channel","PPM","property.type",
                                           "loan.purpose", "seller",
                                           "servicer","def_flag"))

count_factor = sapply(factor_data,
                      function(x) {table(x,exclude = NULL)});count_factor
```

```
## $first.time.homebuyer
## x
##                N       Y    <NA>
##  519840 1716688  233691       0
##
## $insurance
## x
##       0       1    <NA>
## 2149194  321025       0
##
## $number.units
## x
##       1       2       3       4    <NA>
## 2427706   30905    6108    5500       0
##
## $occupancy.status
## x
##       I       O       S    <NA>
##  131684 2212911  125624       0
##
## $channel
## x
##       B       C       R       T    <NA>
##  379333  630606 1186475  273805       0
##
## $PPM
## x
##               N       Y    <NA>
##   23711 2446499       9       0
##
## $property.type
## x
##      CO      CP      LH      MH      PU      SF    <NA>
##  174683    9033    1664    6214  489553 1789072       0
```

```
##
## $loan.purpose
## x
##      C      N      P   <NA>
## 728858 943219 798142      0
##
## $seller
## x
##          AMTRUSTBANK     BANKOFAMERICA,NA BRANCHBANKING&TRUSTC
##                45508               182390               110617
##  CHASEHOMEFINANCELLC     CITIMORTGAGE,INC          COUNTRYWIDE
##               236597               115713               103949
##       FIFTHTHIRDBANK FIRSTHORIZONHOMELOAN FLAGSTARCAPITALMARKE
##                58180                27195                22108
##     GMACMORTGAGE,LLC METLIFEHOMELOANS,ADI         NATLCITYBANK
##                69895                58837                 4622
##       NATLCITYMTGECO        Other sellers          PHHMTGECORP
##                 4540               449465                10562
## PROVIDENTFUNDINGASSO REGIONSBANKDBAREGION SUNTRUSTMORTGAGE,INC
##                66794                 5843                28613
## TAYLOR,BEAN&WHITAKER             USBANKNA WACHOVIAMORTGAGE,FSB
##                88948               195255                12704
## WASHINGTONMUTUALBANK    WELLSFARGOBANK,NA                 <NA>
##                39313               532571                    0
##
## $servicer
## x
##             ALLYBANK          AMTRUSTBANK     BANKOFAMERICA,NA
##                53541                 1732               249631
## BRANCHBANKING&TRUSTC            CENLARFSB         CENTRALMTGECO
##               110617                39592                19366
##     CITIMORTGAGE,INC          COUNTRYWIDE             EVERBANK
##               115521                 1794                 4810
##       FIFTHTHIRDBANK FLAGSTARCAPITALMARKE     GMACMORTGAGE,LLC
##                58180                21352                 5299
## JPMORGANCHASEBANK,NA METLIFEHOMELOANS,ADI NATIONSTARMORTGAGE,L
##               344788                40285                18018
## OCWENLOANSERVICING,L      Other servicers          PHHMTGECORP
##                 7884               460144                10562
##         PNCBANK,NATL PROVIDENTFUNDINGASSO REGIONSBANKDBAREGION
##                 4998                65557                 5844
## SUNTRUSTMORTGAGE,INC TAYLOR,BEAN&WHITAKER             USBANKNA
##                28613                13365               233962
##    WELLSFARGOBANK,NA                 <NA>
##               554764                    0
##
## $def_flag
## x
##   FALSE    TRUE    <NA>
## 2438438   31781       0
```

table above give the number of each classes in each variables.

dummy code categorical variables

```r
relevel_order = function(x){
  tb <- table(x)
  relevel_x <- factor(x,levels = names(tb[order(tb, decreasing = TRUE)]))
  return (relevel_x)
}
```

function for relevel the level's order of each variable by their frequency from high to low

```r
temp = factor_data[,1:8]
name = names(temp)

for (i in 1:8){
  assign(name[i],factor(temp[,i]))
}

first.time.homebuyer = relevel_order(first.time.homebuyer)
dummies1 = model.matrix(~first.time.homebuyer)

insurance = relevel_order(insurance)
dummies2 = model.matrix(~insurance)

number.units = relevel_order(number.units)
dummies3 = model.matrix(~number.units)

occupancy.status = relevel_order(occupancy.status)
dummies4 = model.matrix(~occupancy.status)

channel = relevel_order(channel)
dummies5 = model.matrix(~channel)

PPM = relevel_order(PPM)
dummies6 = model.matrix(~PPM)

property.type = relevel_order(property.type)
dummies7 = model.matrix(~property.type)

loan.purpose = relevel_order(loan.purpose)
dummies8 = model.matrix(~loan.purpose)

dummy_factor_data = cbind(dummies1[,-1],dummies2[,-1],dummies3[,-1],dummies4[,-1],
                          dummies5[,-1],dummies6[,-1],dummies7[,-1],dummies8[,-1])
rm(dummies1, dummies2, dummies3,dummies4,dummies5,dummies6,dummies7,dummies8)
rm(first.time.homebuyer,insurance,number.units,occupancy.status,channel,
   PPM,property.type,loan.purpose)

head(dummy_factor_data)
```

```
##   first.time.homebuyer first.time.homebuyerY   number.units2 number.units3
## 1                    0                     0 0             0             0
## 2                    0                     0 0             0             0
## 3                    0                     0 0             0             0
## 4                    0                     1 1             0             0
## 5                    0                     0 0             0             0
## 6                    0                     0 0             0             0
##   number.units4 occupancy.statusI occupancy.statusS channelC channelB
## 1             0                 0                 0        0        0
## 2             0                 0                 0        0        0
## 3             0                 0                 0        0        0
## 4             0                 0                 0        0        0
## 5             0                 0                 0        0        0
## 6             0                 0                 0        0        0
##   channelT PPM PPMY property.typePU property.typeCO property.typeCP
## 1        0   0    0               0               0               0
## 2        0   0    0               0               0               0
## 3        0   0    0               0               0               0
## 4        0   1    0               0               0               0
## 5        0   0    0               0               0               0
## 6        0   0    0               0               0               0
##   property.typeMH property.typeLH loan.purposeP loan.purposeC
## 1               0               0             0             0
## 2               0               0             0             1
## 3               0               0             0             1
## 4               0               0             1             0
## 5               0               0             0             1
## 6               0               0             0             0
```

numerical variable

score, CLTV, DTI, UPB, LTV, OIR, PPM, orig.loan.term, number.borrowers

```
numerical = subset(D1_removeNA, select=c("score", "CLTV", "DTI", "UPB", "LTV",
                                        "OIR", "orig.loan.term",
                                        "number.borrowers","def_flag"))
```

I translate seller and servicer to numerical variable by using weight of evidence

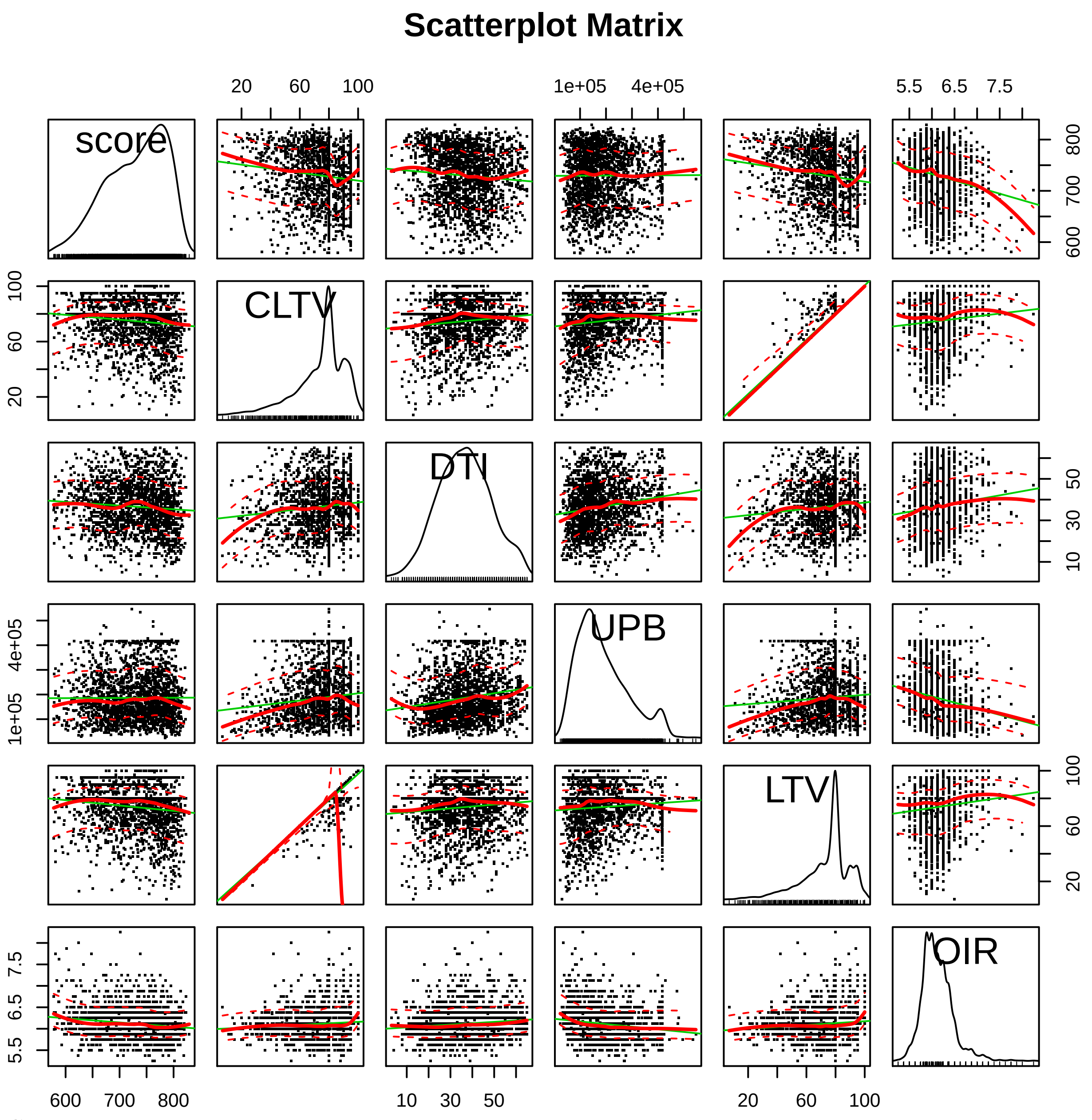
```
woe.tab <- function(x,y) {
  n1 <- sum(y)
  n0 <- sum(1-y)
  nx0n1 <- tapply(1-y,x,sum)*n1
  nx1n0 <- tapply(y,x,sum) *n0
  nx0n1[which(nx0n1==0)]<-n1
  nx1n0[which(nx1n0==0)]<-n0
  log(nx0n1)-log(nx1n0)
}

woe.assign <- function(woetab, x) {
  w<-rep(0,length(x))
  ni<-names(woetab)
  for (i in 1:length(ni)) {
    w[which(x==ni[i])]<-woetab[i]
  }
  w
}
#function woe.tab and woe.assign are writen by Dr Tony

woe_seller = woe.assign(woe.tab(D1_removeNA$seller,D1_removeNA$def_flag),
                        D1_removeNA$seller)

woe_servicer = woe.assign(woe.tab(D1_removeNA$servicer,D1_removeNA$def_flag),
                        D1_removeNA$servicer)
numerical$seller = woe_seller
numerical$servicer = woe_servicer
head(numerical)
```

```
##   score CLTV DTI    UPB LTV   OIR orig.loan.term number.borrowers def_flag
## 1   771   95  61 272000  80 5.875            360                1    FALSE
## 2   729   73  20  87000  73 6.500            360                1    FALSE
## 3   769   59  17  59000  59 6.375            360                1    FALSE
## 4   755  100  28  81000 100 5.875            360                1    FALSE
## 5   760   74  58 165000  74 6.375            360                1    FALSE
## 6   781   80  32 100000  80 6.500            360                1    FALSE
##      seller   servicer
## 1 0.6156186 0.46172759
## 2 0.6156186 0.01796234
## 3 0.6156186 0.01796234
## 4 0.6156186 0.01796234
## 5 0.6156186 0.01796234
## 6 0.6156186 0.01796234
```

some plots for presenting the numerical data

```
scatterplot.matrix(~score+CLTV+DTI+UPB+LTV+OIR, data=numerical[1:2000,],
                   main="Scatterplot Matrix",pch=".")
```

```
## Warning: 'scatterplot.matrix' is deprecated.
## Use 'scatterplotMatrix' instead.
## See help("Deprecated") and help("car-deprecated").
```

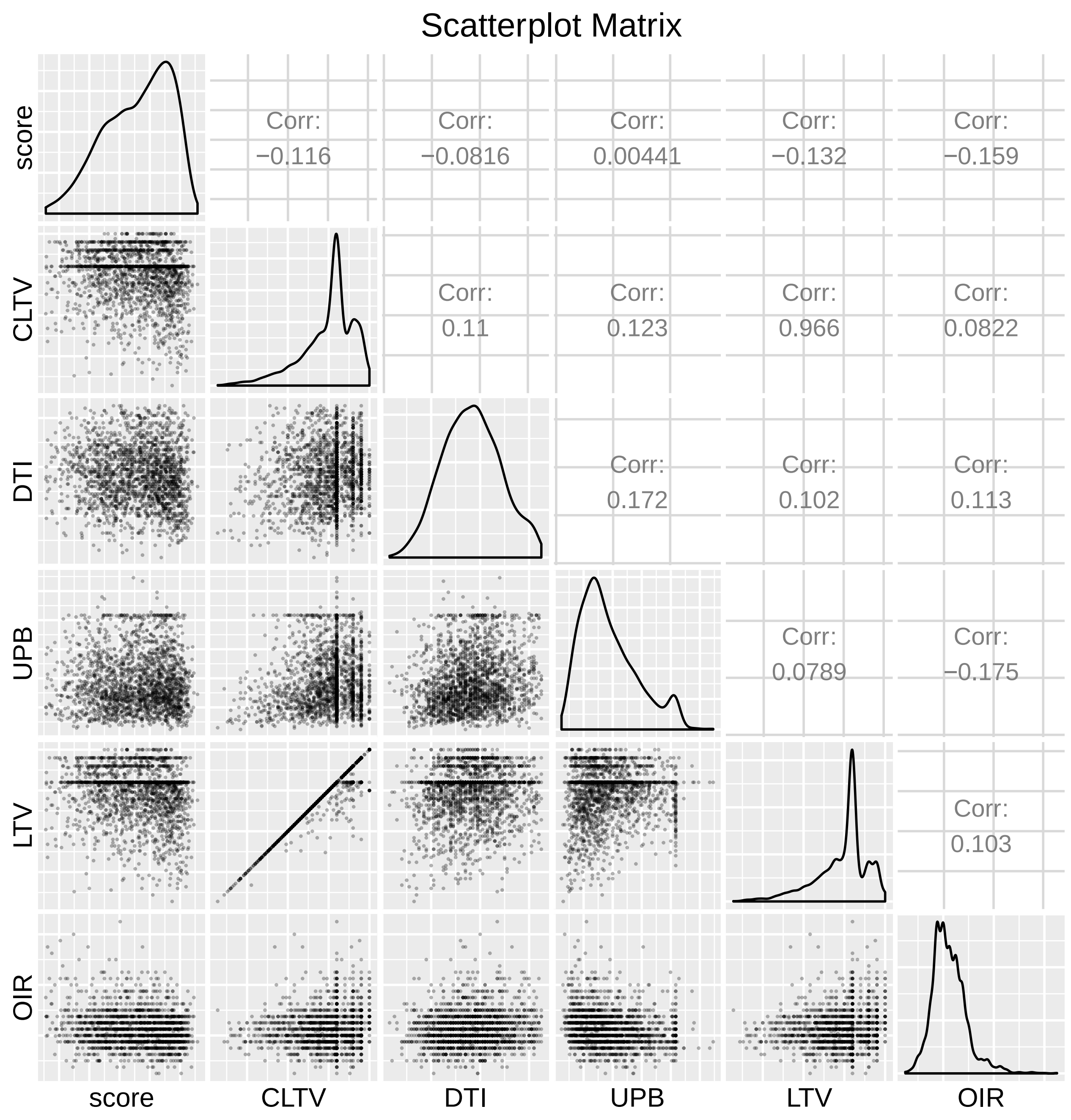


CLTV and LTV are highly correleated

```
library(GGally)
```

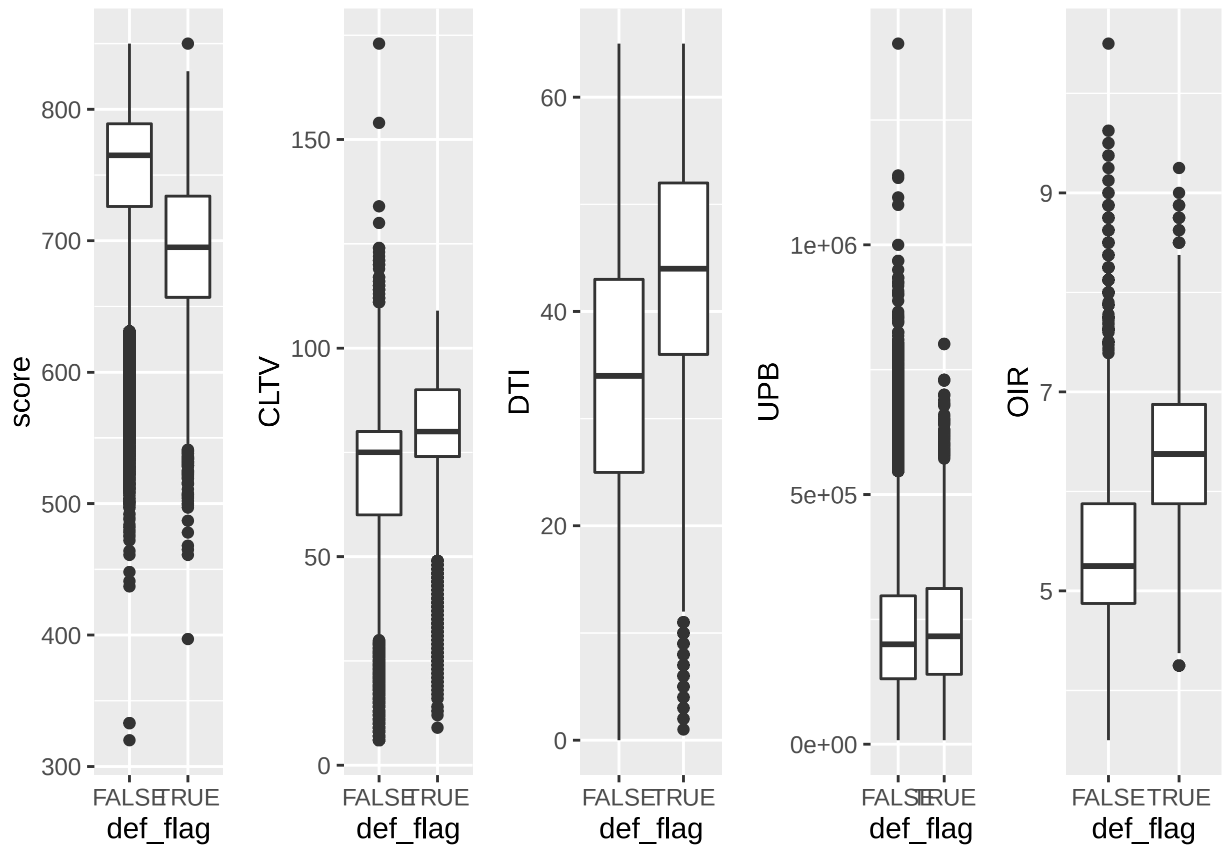
```
## Warning: package 'GGally' was built under R version 3.2.5
```

```
ggpairs(numerical[1:2000,1:6],lower = list(continuous = wrap("points", alpha = 0.3,    size=0.1)),title
```



Below is box plots of score, CLTV, DTI, UPB and OIR by whether default or not

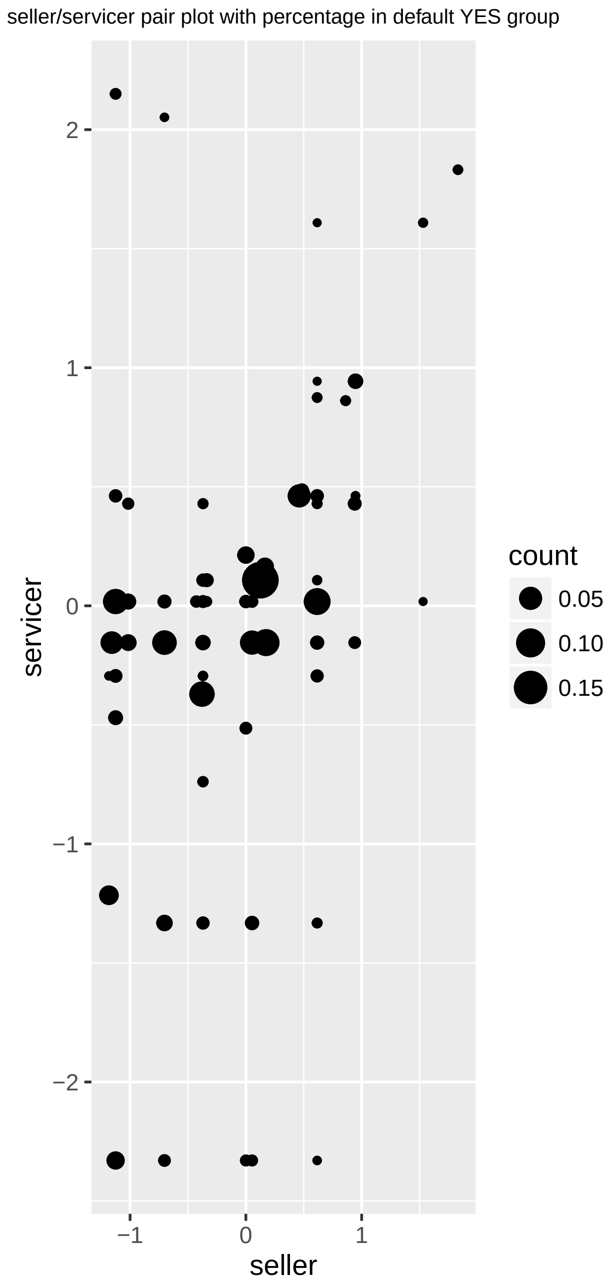
```
p = list()
p[[1]] = ggplot(aes(y = score, x = def_flag), data = numerical) + geom_boxplot()
p[[2]] = ggplot(aes(y = CLTV, x = def_flag), data = numerical) + geom_boxplot()
p[[3]] = ggplot(aes(y = DTI, x = def_flag), data = numerical) + geom_boxplot()
p[[4]] = ggplot(aes(y = UPB, x = def_flag), data = numerical) + geom_boxplot()
p[[5]] = ggplot(aes(y = OIR, x = def_flag), data = numerical) + geom_boxplot()

ggplot2.multiplot(p[[1]],p[[2]],p[[3]],p[[4]],p[[5]], cols=5)
```

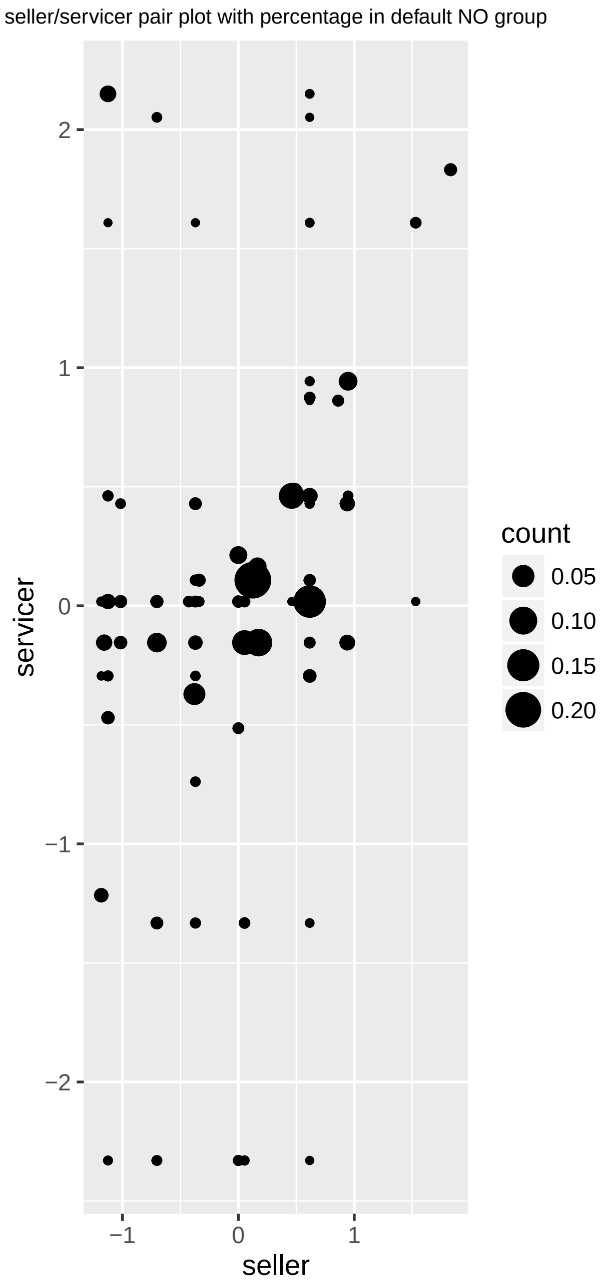


```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:GGally':
##
##     nasa

## The following object is masked from 'package:acs':
##
##     combine

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following object is masked from 'package:MASS':
##
##     select

## The following object is masked from 'package:car':
##
##     recode
## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
pp = list()

a = numerical[which(numerical$def_flag == 1),] %>% group_by(seller,servicer) %>%
summarize(Count = n())

b = numerical[which(numerical$def_flag == 0),] %>% group_by(seller,servicer) %>%
  summarize(Count = n())

a = as.data.frame(a)
names(a)=c('seller','servicer','count')
a$count = (a$count)/sum(a$count)
pp[[1]] <- ggplot(a, aes(seller, servicer)) + geom_point(aes(size = count)) + ggtitle("seller/servicer p


b = as.data.frame(b)
names(b)=c('seller','servicer','count')
b$count = (b$count)/sum(b$count)
pp[[2]] <- ggplot(b, aes(seller, servicer)) + geom_point(aes(size = count))+ggtitle("seller/servicer pa

ggplot2.multiplot(pp[[1]],pp[[2]], cols=2)
```





If we want to check whether seller/servicer pair have some relationships whith possibility of default, we can check the above plot. The dot in the plot represent the pair of seller/servicer, and the size of the certain dot represent $\frac{number\ of\ the\ certain\ pairs}{number\ of\ the\ whole\ pairs}$

There is no clear different pattern between two group.

split the data into train/test datasets

also delete column CLTV

```
data = cbind(numerical,dummy_factor_data)
data = data[,-2]
head(data)
```

```
##   score DTI    UPB LTV   OIR orig.loan.term number.borrowers def_flag
## 1   771  61 272000  80 5.875            360                1    FALSE
## 2   729  20  87000  73 6.500            360                1    FALSE
## 3   769  17  59000  59 6.375            360                1    FALSE
## 4   755  28  81000 100 5.875            360                1    FALSE
## 5   760  58 165000  74 6.375            360                1    FALSE
## 6   781  32 100000  80 6.500            360                1    FALSE
##      seller   servicer first.time.homebuyer first.time.homebuyerY V3
## 1 0.6156186 0.46172759                    0                     0  0
## 2 0.6156186 0.01796234                    0                     0  0
## 3 0.6156186 0.01796234                    0                     0  0
## 4 0.6156186 0.01796234                    0                     1  1
## 5 0.6156186 0.01796234                    0                     0  0
## 6 0.6156186 0.01796234                    0                     0  0
##   number.units2 number.units3 number.units4 occupancy.statusI
## 1             0             0             0                 0
## 2             0             0             0                 0
## 3             0             0             0                 0
## 4             0             0             0                 0
## 5             0             0             0                 0
## 6             0             0             0                 0
##   occupancy.statusS channelC channelB channelT PPM PPMY property.typePU
## 1                 0        0        0        0   0    0               0
## 2                 0        0        0        0   0    0               0
## 3                 0        0        0        0   0    0               0
## 4                 0        0        0        0   1    0               0
## 5                 0        0        0        0   0    0               0
## 6                 0        0        0        0   0    0               0
##   property.typeCO property.typeCP property.typeMH property.typeLH
## 1               0               0               0               0
## 2               0               0               0               0
## 3               0               0               0               0
## 4               0               0               0               0
## 5               0               0               0               0
## 6               0               0               0               0
##   loan.purposeP loan.purposeC
## 1             0             0
## 2             0             1
## 3             0             1
## 4             1             0
## 5             0             1
## 6             0             0
```

```
sample_index = sample(1:nrow(data), floor(nrow(data)/10), replace=FALSE)
train <- data[sample_index,]
test <- data[-sample_index,]
```

glm function

fit model by using "glm"

default

$$\alpha = 1$$

which means losso

```
glm.fit = glm(def_flag ~ . , data=train, family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = def_flag ~ ., family = binomial, data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6278  -0.1305  -0.0745  -0.0449   4.1321
##
## Coefficients: (1 not defined because of singularities)
##                         Estimate Std. Error z value Pr(>|z|)
## (Intercept)           -4.003e+00  7.747e+00  -0.517 0.605362
## score                 -1.359e-02  3.979e-04 -34.163  < 2e-16 ***
## DTI                    3.028e-02  1.691e-03  17.903  < 2e-16 ***
## UPB                    3.125e-06  1.772e-07  17.636  < 2e-16 ***
## LTV                    2.905e-02  2.183e-03  13.307  < 2e-16 ***
## OIR                    8.811e-01  3.376e-02  26.101  < 2e-16 ***
## orig.loan.term         3.869e-03  2.148e-02   0.180 0.857087
## number.borrowers      -9.526e-01  4.192e-02 -22.720  < 2e-16 ***
## seller                -9.003e-02  3.512e-02  -2.563 0.010365 *
## servicer              -3.977e-01  3.243e-02 -12.263  < 2e-16 ***
## first.time.homebuyer  -1.813e-03  5.309e-02  -0.034 0.972755
## first.time.homebuyerY  9.244e-02  6.717e-02   1.376 0.168761
## V3                     3.355e-01  5.775e-02   5.809 6.28e-09 ***
## number.units2          3.491e-01  1.027e-01   3.399 0.000675 ***
## number.units3          3.390e-01  2.254e-01   1.504 0.132657
## number.units4         -4.582e-01  3.500e-01  -1.309 0.190421
## occupancy.statusI      4.340e-01  6.902e-02   6.289 3.20e-10 ***
## occupancy.statusS      4.715e-01  8.948e-02   5.269 1.37e-07 ***
## channelC               2.859e-01  5.506e-02   5.194 2.06e-07 ***
## channelB               4.274e-01  5.896e-02   7.248 4.22e-13 ***
## channelT               4.864e-01  5.043e-02   9.644  < 2e-16 ***
## PPM                    8.179e-01  8.644e-02   9.463  < 2e-16 ***
## PPMY                          NA         NA      NA       NA
## property.typePU       -1.723e-01  5.566e-02  -3.095 0.001968 **
## property.typeCO        2.084e-01  6.493e-02   3.210 0.001329 **
## property.typeCP       -3.448e-01  4.546e-01  -0.759 0.448098
## property.typeMH       -2.314e-01  4.210e-01  -0.550 0.582525
## property.typeLH       -2.859e-01  7.776e-01  -0.368 0.713158
## loan.purposeP         -4.725e-01  5.614e-02  -8.416  < 2e-16 ***
## loan.purposeC          8.382e-02  5.031e-02   1.666 0.095713 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 33932  on 247020  degrees of freedom
## Residual deviance: 25174  on 246992  degrees of freedom
## AIC: 25232
##
## Number of Fisher Scoring iterations: 8
```

why PPMY is NA?

```
table(D1_removeNA$PPM,exclude = NULL)
```

```
##
##                 N       Y    <NA>
##   23711 2446499       9       0
```

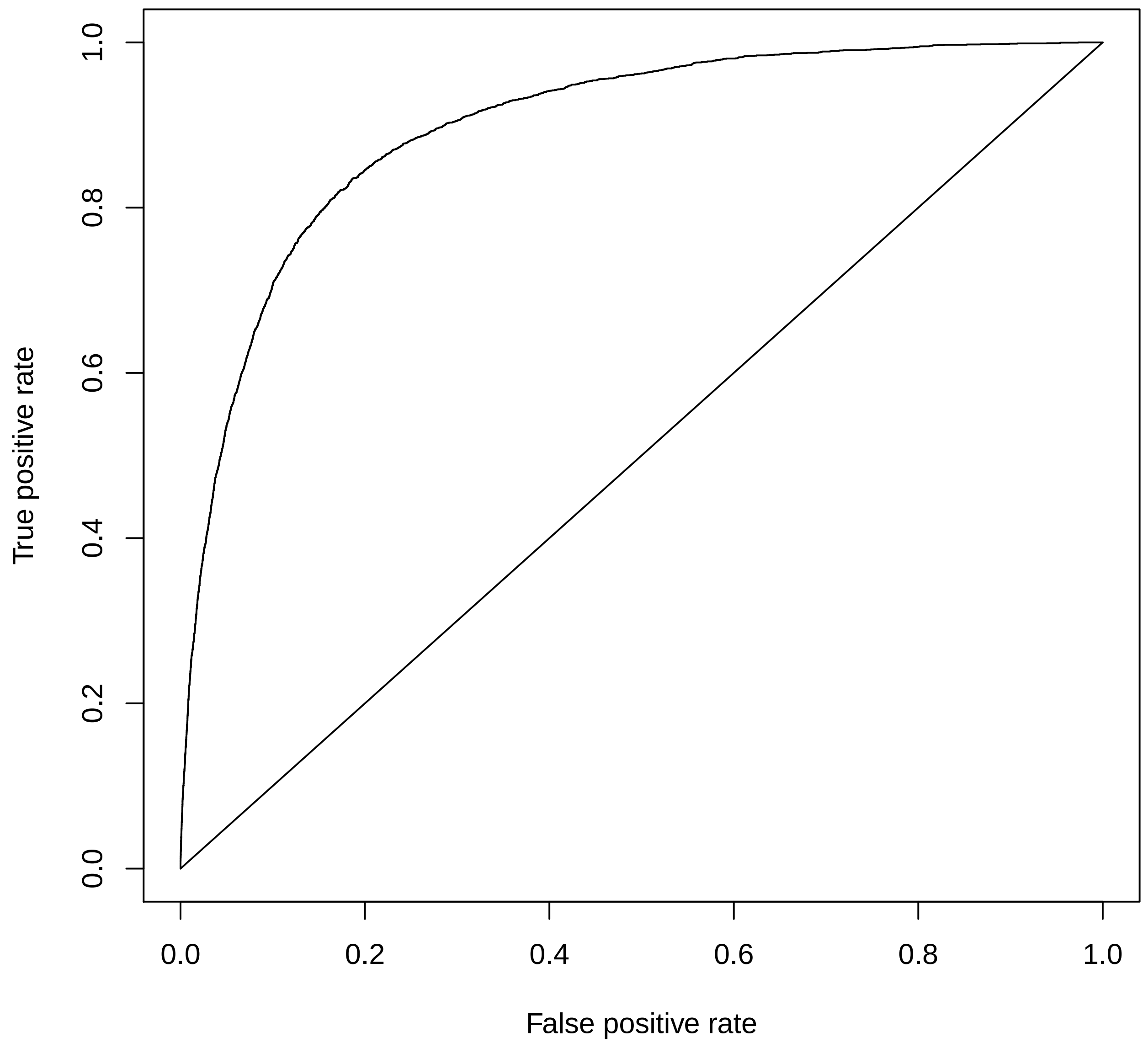
```
coef(glm.fit)
```

```
##           (Intercept)                 score                   DTI
##         -4.002795e+00         -1.359333e-02          3.028086e-02
##                   UPB                   LTV                   OIR
##          3.125184e-06          2.904836e-02          8.811040e-01
##       orig.loan.term     number.borrowers                seller
##          3.868962e-03         -9.525535e-01         -9.002603e-02
##             servicer  first.time.homebuyer first.time.homebuyerY
##         -3.977145e-01         -1.813377e-03          9.244493e-02
##                    V3         number.units2         number.units3
##          3.354741e-01          3.491129e-01          3.389510e-01
##         number.units4      occupancy.statusI     occupancy.statusS
##         -4.582214e-01          4.340426e-01          4.714949e-01
##              channelC              channelB              channelT
##          2.859483e-01          4.273533e-01          4.863806e-01
##                   PPM                  PPMY       property.typePU
##          8.179352e-01                    NA         -1.722731e-01
##       property.typeCO       property.typeCP       property.typeMH
##          2.084104e-01         -3.448226e-01         -2.314356e-01
##       property.typeLH          loan.purposeP          loan.purposeC
##         -2.858597e-01         -4.724519e-01          8.381919e-02
```

```
glm.probs=predict(glm.fit,type="response")
pr <- prediction(glm.probs, train$def_flag)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]];auc
```

```
## [1] 0.8978686
```

```
plot(prf);lines(x = c(0,1), y = c(0,1))
```



On train data, the AUC is 0.8978686

the model performence on test data

```
fitted.results <- predict(glm.fit,newdata=test,
                          type='response')
```

```
## Warning in predict.lm(object, newdata, se.fit, scale = 1, type =
## ifelse(type == : prediction from a rank-deficient fit may be misleading
```

```
fitted.outputs <- ifelse(fitted.results > 0.5, 1, 0)
misClasificError <- mean(fitted.outputs != test$def_flag,na.omit="TRUE")
misClasificError
```

```
## [1] 0.01293092
```

If I set the threshold to 0.5, the mis-classify rate will be 0.0129309 so let's try different threshold from 0.1 to 0.9

```
threshold = function(x){
  fitted.outputs <- ifelse(fitted.results > x, 1, 0)
  misClasificError <- mean(fitted.outputs != test$def_flag,na.omit="TRUE")
  return (misClasificError)
}

rate = lapply(seq(0.1,0.9,0.1),threshold)
unlist(rate)
```

```
## [1] 0.02889306 0.01677358 0.01404598 0.01319765 0.01293092 0.01287560
## [7] 0.01286705 0.01286975 0.01286705
```

```
fitted.outputs <- ifelse(fitted.results > 0.5, 1, 0)
table(fitted.outputs,test$def_flag)
```

```
##
## fitted.outputs   FALSE    TRUE
##              0 2193930   28089
##              1     659     520
```

```
fitted.outputs <- ifelse(fitted.results > 0.1, 1, 0)
table(fitted.outputs,test$def_flag)
```

```
##
## fitted.outputs   FALSE    TRUE
##              0 2149118   18764
##              1   45471    9845
```

above is typeI and typrII error table with threshold 0.5 and 0.1 respectively

we value more on typeII error, so lets place more weight on typeII error, let's say typeII:typeI = 3:1 here

```
weighterror = function(threshold){
  fitted.outputs = ifelse(fitted.results > threshold, 1, 0)
  errortable = table(fitted.outputs,test$def_flag)
  weight_error = (3*errortable[1,2]+errortable[2,1])/sum(errortable)
  return(weight_error)
}
weighter = lapply(seq(0.1,0.9,0.01),weighterror)
unlist(weighter)
```

```
##  [1] 0.04577325 0.04409549 0.04282075 0.04173852 0.04092258 0.04031040
##  [7] 0.03978008 0.03937796 0.03902127 0.03874329 0.03850714 0.03836365
## [13] 0.03822242 0.03806813 0.03802450 0.03801326 0.03792645 0.03788911
## [19] 0.03789091 0.03784503 0.03784863 0.03784863 0.03785403 0.03786527
## [25] 0.03790261 0.03792015 0.03791205 0.03790845 0.03793769 0.03796018
## [31] 0.03796468 0.03800156 0.03802495 0.03805419 0.03806948 0.03808388
## [37] 0.03811401 0.03813381 0.03814820 0.03816529 0.03819993 0.03823591
## [43] 0.03825390 0.03827504 0.03830338 0.03831957 0.03833487 0.03834656
## [49] 0.03837670 0.03840099 0.03842258 0.03844687 0.03845362 0.03846936
## [55] 0.03847790 0.03849005 0.03850309 0.03851479 0.03852333 0.03853188
## [61] 0.03853818 0.03854672 0.03855392 0.03856202 0.03856786 0.03857236
## [67] 0.03857236 0.03858046 0.03858271 0.03859080 0.03859395 0.03859620
## [73] 0.03859620 0.03859620 0.03859665 0.03859665 0.03859575 0.03859800
## [79] 0.03859890 0.03860115 0.03860115
```

```
min(unlist(weighter))
```

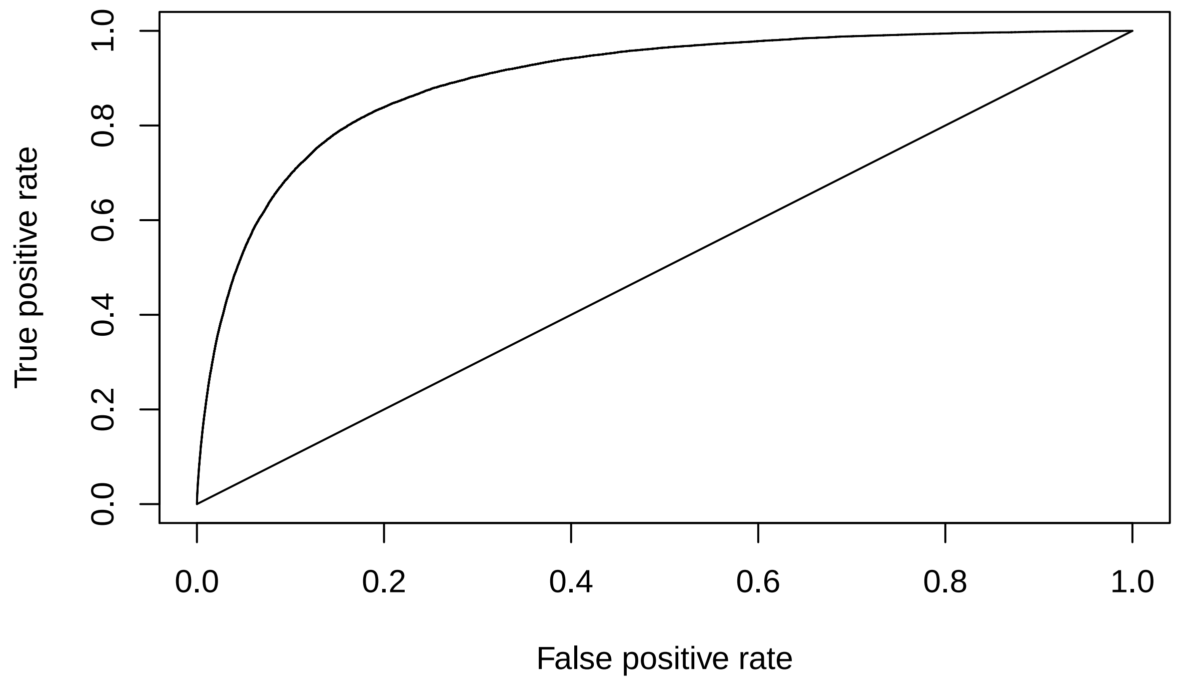
```
## [1] 0.03784503
```

```
which.min(unlist(weighter))
```

```
## [1] 20
```

we tried the threshold from 0.1 to 0.9 by 0.01 with weghtted typeI and typeII error, here the best threhold is 0.29, and the weighted error rate is 0.037845

```
pr <- prediction(fitted.results, test$def_flag)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
plot(prf);lines(x = c(0,1), y = c(0,1))
```



```
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
## [1] 0.8963456
```

AUC value is 0.8963456

above penalty is lasso (defult alpha =1)

let's tune alpha and lamda by 10 fold Cross Validation

using glmnet with elasticnet penalty

```
names(train)
```

```
##  [1] "score"                "DTI"
##  [3] "UPB"                  "LTV"
##  [5] "OIR"                  "orig.loan.term"
##  [7] "number.borrowers"     "def_flag"
##  [9] "seller"               "servicer"
## [11] "first.time.homebuyer" "first.time.homebuyerY"
## [13] "V3"                   "number.units2"
## [15] "number.units3"        "number.units4"
## [17] "occupancy.statusI"    "occupancy.statusS"
## [19] "channelC"             "channelB"
## [21] "channelT"             "PPM"
## [23] "PPMY"                 "property.typePU"
## [25] "property.typeCO"      "property.typeCP"
## [27] "property.typeMH"      "property.typeLH"
## [29] "loan.purposeP"        "loan.purposeC"
```

```
x = train[,-8]
x = as.matrix(x)
y = train[,8]
```

here we choose AUC as measure methods in cross validation

above penalty is lasso (defult alpha =1)

let's also tune alpha here, makes penatly become elastic net

below makes a alpha and lamda grid with alpha density 0.1 and lamda density 0.0001(defaul setting in cv.glmnet),tune on a 10 fold cross validation, measure is AUC

```
alphaslist<-seq(0,1,by=0.1)

temp_function = function(i){
    cvfit = cv.glmnet(x, y, family='binomial',type.measure = "auc",alpha = i)
    fitted.results = predict(cvfit, newx = as.matrix(test[,-8]), s = "lambda.min",
                             type = "response")
    pr <- prediction(fitted.results, test$def_flag)
    auc <- performance(pr, measure = "auc")
    auc <- auc@y.values[[1]]
    return(c(auc,i))
}
```

```r
temp = lapply(alphaslist, temp_function)
```

```r
temp = cbind(unlist(temp)[seq(2, length(unlist(temp)), 2)],
             unlist(temp)[seq(1, length(unlist(temp)), 2)])
temp = as.data.frame(temp)
names(temp)=c('alpha','auc')
temp
```

```
##    alpha       auc
## 1    0.0 0.8963361
## 2    0.1 0.8965033
## 3    0.2 0.8964969
## 4    0.3 0.8964867
## 5    0.4 0.8964781
## 6    0.5 0.8964766
## 7    0.6 0.8964675
## 8    0.7 0.8964650
## 9    0.8 0.8964625
## 10   0.9 0.8964398
## 11   1.0 0.8964383
```

```r
max(temp$auc)
```

```
## [1] 0.8965033
```

```r
temp$alpha[which.max((temp$auc))]
```

```
## [1] 0.1
```

so choose

$$\alpha = 0.1$$

```r
cvfit = cv.glmnet(x, y, family='binomial',type.measure = "auc",alpha = temp$alpha[which.max((temp$auc))
cvfit$lambda.min
```

```
## [1] 0.0008430041
```

```r
coef(cvfit, s = "lambda.min")
```

```
## 30 x 1 sparse Matrix of class "dgCMatrix"
##                                 1
## (Intercept)         -2.375739e+00
## score               -1.289518e-02
## DTI                  2.868704e-02
## UPB                  2.697184e-06
## LTV                  2.299544e-02
## OIR                  8.479481e-01
## orig.loan.term       .
## number.borrowers    -8.588850e-01
## seller                -1.178283e-01
## servicer              -3.832059e-01
## first.time.homebuyer   .
## first.time.homebuyerY  2.069852e-02
## V3                     3.885031e-01
## number.units2          3.631389e-01
## number.units3          3.318807e-01
## number.units4         -2.831241e-01
## occupancy.statusI      3.518279e-01
## occupancy.statusS      3.462805e-01
## channelC               1.979036e-01
## channelB               3.483242e-01
## channelT               4.519228e-01
## PPM                    8.223382e-01
## PPMY                   .
## property.typePU       -1.387918e-01
## property.typeCO        1.900734e-01
## property.typeCP       -1.543311e-01
## property.typeMH       -1.229378e-01
## property.typeLH        .
## loan.purposeP         -3.586880e-01
## loan.purposeC          7.503881e-02
```

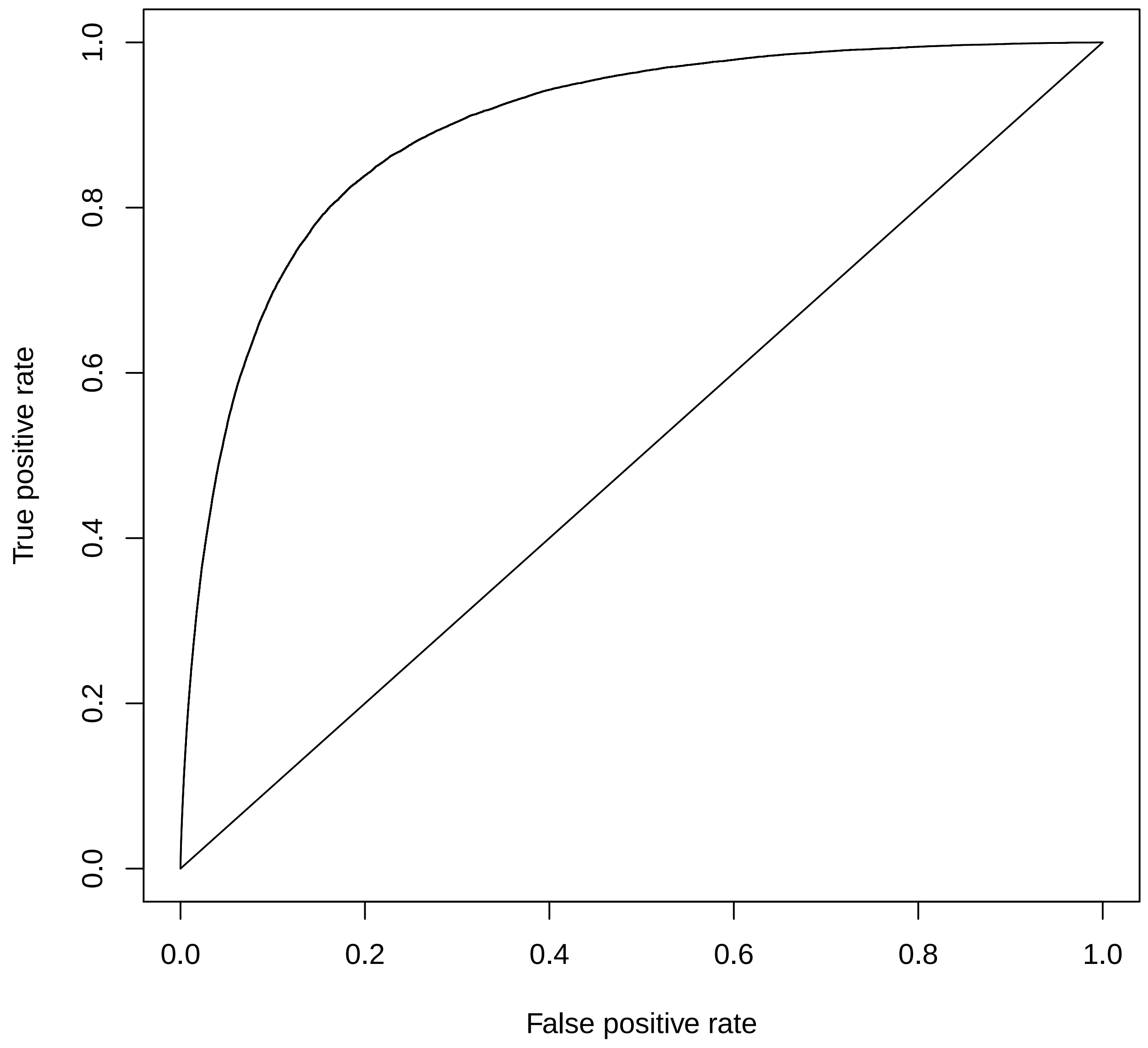
```
pre = predict(cvfit, newx = as.matrix(test[,-8]), s = "lambda.min", type = "class")
```

the performance on test data

```
fitted.results = predict(cvfit, newx = as.matrix(test[,-8]), s = "lambda.min",
                         type = "response")
pr <- prediction(fitted.results, test$def_flag)
prf <- performance(pr, measure = "tpr", x.measure = "fpr")
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]];auc
```

```
## [1] 0.8964943
```

```
plot(prf);lines(x = c(0,1), y = c(0,1))
```



auc equal to 0.8964943, which is a little bit better than previous model.

let's also try the weighted error rate

```
weighter = lapply(seq(0.1,0.9,0.01),weighterror)
min(unlist(weighter))
```

```
## [1] 0.03785628
```

```
which.min(unlist(weighter))
```

```
## [1] 19
```

the lowest weighted error rate is 0.0378563, when choose 0.28 as threshold

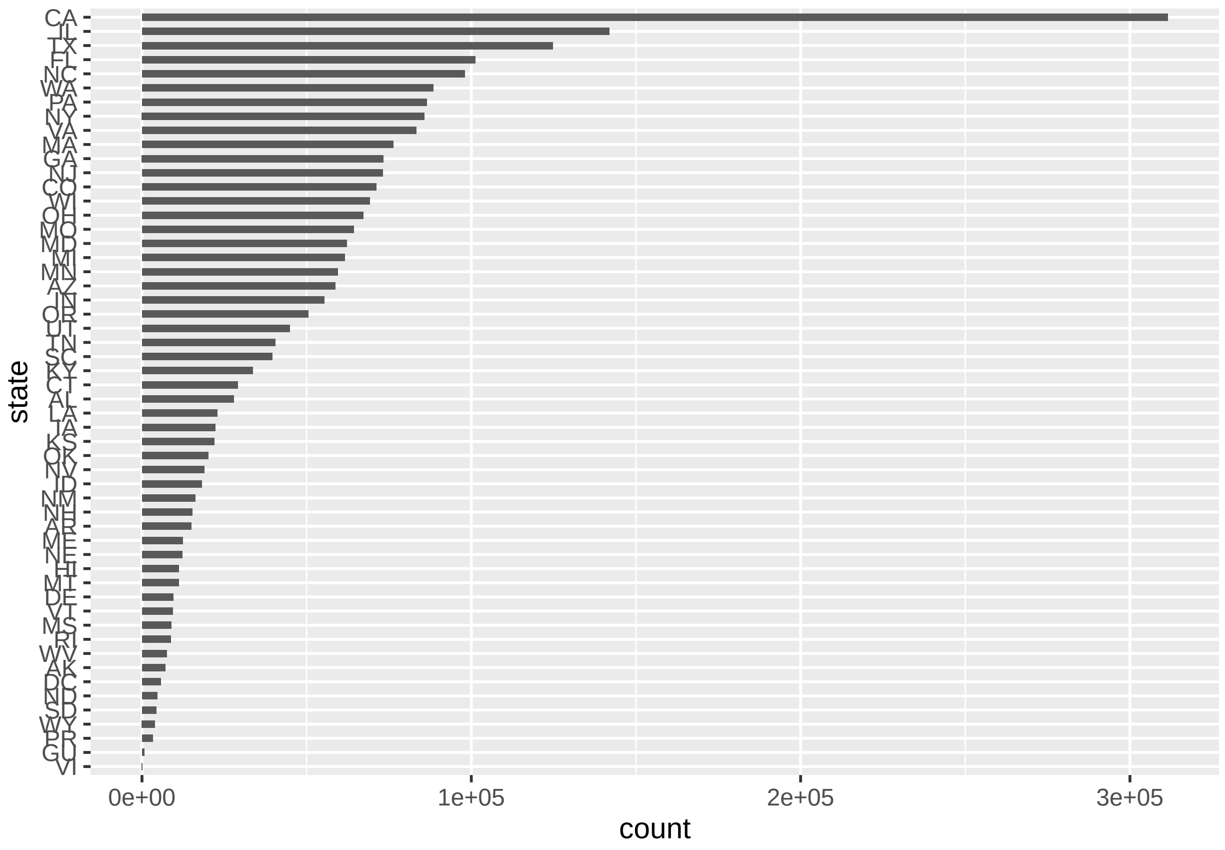
```
coeftable = cbind(coef(cvfit, s = "lambda.min"),coef(glm.fit));coeftable
```

```
## 30 x 2 sparse Matrix of class "dgCMatrix"
##                                  1
## (Intercept)           -2.375739e+00 -4.002795e+00
## score                 -1.289518e-02 -1.359333e-02
## DTI                    2.868704e-02  3.028086e-02
## UPB                    2.697184e-06  3.125184e-06
## LTV                    2.299544e-02  2.904836e-02
## OIR                    8.479481e-01  8.811040e-01
## orig.loan.term         .              3.868962e-03
## number.borrowers      -8.588850e-01 -9.525535e-01
## seller                -1.178283e-01 -9.002603e-02
## servicer              -3.832059e-01 -3.977145e-01
## first.time.homebuyer   .             -1.813377e-03
## first.time.homebuyerY  2.069852e-02  9.244493e-02
## V3                     3.885031e-01  3.354741e-01
## number.units2          3.631389e-01  3.491129e-01
## number.units3          3.318807e-01  3.389510e-01
## number.units4         -2.831241e-01 -4.582214e-01
## occupancy.statusI      3.518279e-01  4.340426e-01
## occupancy.statusS      3.462805e-01  4.714949e-01
## channelC               1.979036e-01  2.859483e-01
## channelB               3.483242e-01  4.273533e-01
## channelT               4.519228e-01  4.863806e-01
## PPM                    8.223382e-01  8.179352e-01
## PPMY                   .                        NA
## property.typePU       -1.387918e-01 -1.722731e-01
## property.typeCO        1.900734e-01  2.084104e-01
## property.typeCP       -1.543311e-01 -3.448226e-01
## property.typeMH       -1.229378e-01 -2.314356e-01
## property.typeLH        .             -2.858597e-01
## loan.purposeP         -3.586880e-01 -4.724519e-01
## loan.purposeC          7.503881e-02  8.381919e-02
```

above shows the cofficient, left column is logistic model with elasticnet penalty, right column is logistic midel.

how about focus on a certain state?

```
D_state = D1[,-c(1,3,6,5,17,20,26)]
D_state = D_state[complete.cases(D_state),]
state = D_state$property.state
theTable = as.data.frame(state)

theTable <- within(theTable,
                   state <- factor(state,
                                   levels=names(sort(table(state),
                                                     increasing=TRUE))))

m <- ggplot(theTable, aes(x=state))
m + stat_count(width = 0.5) + coord_flip()
```



In this data set, california occupy more than 10% data. if we just use the data from california to fit a model, will this be a good model for Taxes and the whole U.S.? (if california data is biased data set, I guess maybe the model will not perform very well)

build the model on CA

```
ca = data[(state=="CA"),]
tx = data[(state=="TX"),]
except_ca = data[(state!="CA"),]
```

```
x = ca[,-8]
x = as.matrix(x)
y = ca[,8]
cvfit = cv.glmnet(x, y, family='binomial',type.measure = "auc")
coef(cvfit, s = "lambda.min")
```

```
## 30 x 1 sparse Matrix of class "dgCMatrix"
##                                    1
## (Intercept)            -5.096218e+00
## score                  -1.173278e-02
## DTI                     3.084258e-02
## UPB                     1.203499e-06
## LTV                     4.420271e-02
## OIR                     1.018554e+00
## orig.loan.term         .
## number.borrowers      -7.877021e-01
## seller                  .
## servicer              -3.162420e-01
## first.time.homebuyer  -1.545930e-01
## first.time.homebuyerY -1.409340e-01
## V3                     4.494251e-01
## number.units2          3.105876e-01
## number.units3           .
## number.units4           .
## occupancy.statusI     -6.321794e-02
## occupancy.statusS       .
## channelC               1.395659e-01
## channelB              -6.897772e-02
## channelT               5.504631e-01
## PPM                    9.176358e-01
## PPMY                    .
## property.typePU       -1.345672e-01
## property.typeCO       -1.243928e-01
## property.typeCP         .
## property.typeMH       -5.031380e-02
## property.typeLH        2.428347e+00
## loan.purposeP         -5.776119e-01
## loan.purposeC          1.398406e-01
```

we can see more coefficient are shrinkage to 0.

calculate the auc on CA, TX and non-CA data sets

```
auc_calucator = function(model, test_data_x, test_data_y) {
  fitted.results = predict(model, newx = test_data_x, s = "lambda.min",
                           type = "response")
  pr <- prediction(fitted.results, test_data_y)
  prf <- performance(pr, measure = "tpr", x.measure = "fpr")
  auc <- performance(pr, measure = "auc")
  auc <- auc@y.values[[1]]
  return(auc)
}

auc_ca = auc_calucator(cvfit, x, y);auc_ca
```

```
## [1] 0.9064395
```

```
auc_tx = auc_calucator(cvfit, as.matrix(tx[,-8]), tx[,8]);auc_tx
```

```
## [1] 0.878791
```

```
auc_nonca = auc_calucator(cvfit, as.matrix(except_ca[,-8]), except_ca[,8]);auc_nonca
```

```
## [1] 0.8890395
```

The auc value on California data itself is 0.9064395, on Taxes data is 0.878791, on all non-California data is 0.8890395. We could guess if we fit the model based on every state itself, the 54 models performance on their own state maybe better than we fit a model on the whole dataset.

split the data set by state

```
data = cbind(data,state)
out <- split( data , f = data$state )
head(out$CA)
```

```
##     score DTI    UPB LTV   OIR orig.loan.term number.borrowers def_flag
## 89    795  39 252000  80 6.250            360                2    FALSE
## 91    733  30 150000  38 6.375            360                2    FALSE
## 183   664  49 255000  85 6.000            360                2    FALSE
## 198   767  37 270000  77 6.375            360                2    FALSE
## 202   680  48 265000  69 6.250            360                2    FALSE
## 245   690  12 140000  36 6.500            360                2    FALSE
##        seller   servicer first.time.homebuyer first.time.homebuyerY V3
## 89  0.6156186 0.01796234                    0                     0  0
## 91  0.6156186 0.01796234                    0                     0  0
## 183 0.6156186 0.01796234                    0                     0  1
## 198 0.6156186 0.01796234                    0                     0  0
## 202 0.6156186 0.01796234                    0                     0  0
## 245 0.6156186 0.46172759                    0                     0  0
##     number.units2 number.units3 number.units4 occupancy.statusI
## 89              0             0             0                 0
## 91              0             0             0                 0
## 183             0             0             0                 0
## 198             0             0             0                 0
## 202             0             0             0                 0
## 245             0             0             0                 0
##     occupancy.statusS channelC channelB channelT PPM PPMY property.typePU
## 89                  0        0        0        0   0    0               1
## 91                  0        0        0        0   0    0               0
## 183                 0        0        0        0   0    0               0
## 198                 0        0        0        0   0    0               0
## 202                 0        0        0        0   0    0               0
## 245                 0        0        0        0   0    0               0
##     property.typeCO property.typeCP property.typeMH property.typeLH
## 89                0               0               0               0
## 91                0               0               0               0
## 183               0               0               0               0
## 198               0               0               0               0
## 202               0               0               0               0
## 245               0               0               0               0
##     loan.purposeP loan.purposeC state
## 89              0             0    CA
## 91              0             1    CA
## 183             0             1    CA
## 198             0             1    CA
## 202             0             1    CA
## 245             0             1    CA
```

try to get a lsit of CV.FIT in different state

```r
my.fit.function = function(state_data){
  d = state_data[,-ncol(state_data)]
  #x = d[,-8]
  #x = as.matrix(x)
  #y = d[,8]
  #glm.fit = cv.glmnet(x, y, family='binomial',type.measure = "auc",nfolds = 3)
  glm.fit = glm(def_flag ~ . , data = d, family=binomial)
  return(glm.fit)
}

state.cv.fit = lapply(out,my.fit.function)
```

```r
state.cv.fit[[1]]
```

```
##
## Call:  glm(formula = def_flag ~ ., family = binomial, data = d)
##
## Coefficients:
##           (Intercept)                  score                    DTI
##            -9.597e+01             -1.859e-02              5.834e-02
##                   UPB                    LTV                    OIR
##             2.989e-06              7.411e-02              3.411e-01
##        orig.loan.term      number.borrowers                 seller
##             2.647e-01             -1.160e+00             -5.508e-02
##              servicer   first.time.homebuyer  first.time.homebuyerY
##            -9.241e-01             -1.500e-01              1.608e-01
##                    V3          number.units2          number.units3
##            -5.667e-01             -1.018e+00             -1.491e+01
##         number.units4      occupancy.statusI      occupancy.statusS
##            -1.515e+01              1.219e+00              1.125e+00
##              channelC               channelB               channelT
##             4.906e-01              1.947e-02              2.682e-01
##                   PPM                   PPMY         property.typePU
##            -1.463e+01                     NA             -1.147e+00
##       property.typeCO        property.typeCP         property.typeMH
##             3.075e-01                     NA             -1.284e+01
##       property.typeLH         loan.purposeP          loan.purposeC
##                    NA             -8.913e-01             -1.864e-01
##
## Degrees of Freedom: 7013 Total (i.e. Null);  6987 Residual
## Null Deviance:       482.8
## Residual Deviance: 358.9     AIC: 412.9
```

```r
auc_record = function(fit,data){
  fitted.results <- predict(fit,newdata=data[,-ncol(data)],
                            type='response')
  pr <- prediction(fitted.results, data$def_flag)
  auc <- performance(pr, measure = "auc")
  auc <- auc@y.values[[1]]
  return(auc)
}

auc_record(fit=state.cv.fit[[1]], data = out[[2]])
```

```
## [1] 0.8007798
```

```
auc_matrix=matrix(0,nrow=54,ncol=54)
for (i in 1:54){
  for (j in 1:54){
    auc_matrix[i,j] = auc_record(fit=state.cv.fit[[i]],out[[j]])
  }
}
row.names(auc_matrix)=names(out)
colnames(auc_matrix)=names(out)

auc_matrix
```

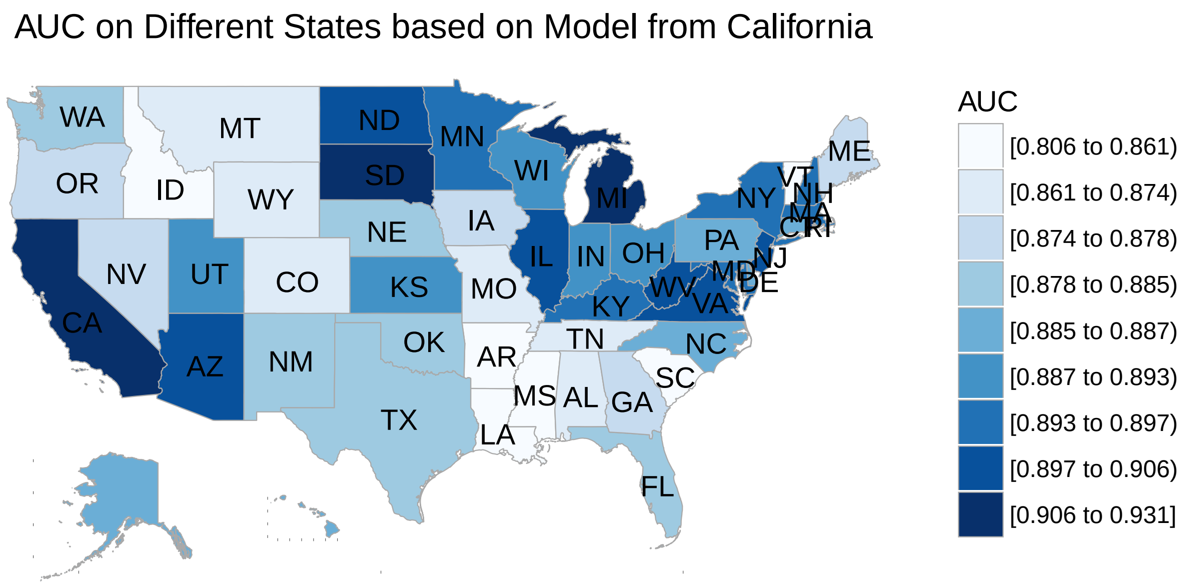
```
##           AK        AL        AR        AZ        CA        CO        CT
## AK 0.9219263 0.8007798 0.7700495 0.8015581 0.8297301 0.7963640 0.7952608
## AL 0.8739932 0.8779037 0.8450884 0.8815739 0.8477174 0.8648245 0.7853485
## AR 0.7873909 0.8430356 0.8745673 0.8222226 0.8134705 0.8012586 0.7903204
## AZ 0.8874148 0.8586036 0.8322304 0.9034422 0.8975656 0.8717712 0.8765139
## CA 0.8847790 0.8611609 0.8399194 0.8989818 0.9061680 0.8678885 0.8854942
## CO 0.8899366 0.8679937 0.8538919 0.8930764 0.8868382 0.8897425 0.8810575
## CT 0.8938737 0.8621612 0.8438746 0.8842797 0.8961348 0.8709227 0.9013729
## DC 0.8028527 0.7774411 0.7542499 0.7616720 0.7955543 0.7158753 0.7648084
## DE 0.8738572 0.8513934 0.8316917 0.8677028 0.8399269 0.8583329 0.8208045
## FL 0.8742799 0.8569984 0.8321305 0.8940018 0.8863190 0.8709459 0.8689601
## GA 0.8815072 0.8686097 0.8520395 0.8884216 0.8871027 0.8738330 0.8561543
## GU 0.5161364 0.5046303 0.5184030 0.5023613 0.5180442 0.5112844 0.5247655
## HI 0.8877530 0.8462591 0.8405742 0.8800090 0.8771910 0.8689837 0.8536421
## IA 0.8776951 0.8388818 0.8358064 0.8800502 0.8615663 0.8606348 0.8402302
## ID 0.8816910 0.8441529 0.8327965 0.8773558 0.8421205 0.8601948 0.8200936
## IL 0.8878302 0.8622843 0.8422429 0.8934298 0.8924227 0.8692136 0.8820239
## IN 0.9015826 0.8613608 0.8481085 0.8791207 0.8748532 0.8693056 0.8534144
## KS 0.7863432 0.8390674 0.8410167 0.8174953 0.8133262 0.8047092 0.7992539
## KY 0.9086040 0.8586896 0.8490049 0.8811853 0.8851437 0.8677385 0.8620854
## LA 0.8874920 0.8669572 0.8575157 0.8831380 0.8713658 0.8805421 0.8456643
## MA 0.8781325 0.7986429 0.7831122 0.7783224 0.8506121 0.7545715 0.8865346
## MD 0.8956603 0.8593250 0.8418406 0.8883411 0.8968066 0.8739576 0.8876476
## ME 0.8525062 0.7666815 0.7628606 0.7656803 0.8249996 0.7382147 0.8665775
## MI 0.8832423 0.8554378 0.8355282 0.8811753 0.8876508 0.8507758 0.8783752
## MN 0.8906351 0.8638075 0.8481762 0.8942072 0.8883042 0.8720162 0.8803076
## MO 0.9022333 0.8645506 0.8490119 0.8853610 0.8889019 0.8677197 0.8583580
## MS 0.8816028 0.8645956 0.8353268 0.8652378 0.8428459 0.8676154 0.8307055
## MT 0.8938848 0.8426282 0.8357000 0.8691690 0.8512708 0.8568625 0.8371257
## NC 0.8948700 0.8663121 0.8550209 0.8920137 0.8842579 0.8827507 0.8839518
## ND 0.6638691 0.5939751 0.5959261 0.6668482 0.6507688 0.6152757 0.6356862
## NE 0.7324364 0.8053257 0.8248620 0.7997654 0.7983723 0.7764426 0.7805516
## NH 0.8952854 0.8516433 0.8360669 0.8799318 0.8788444 0.8545194 0.8597655
## NJ 0.8884551 0.8570852 0.8418003 0.8861186 0.8943887 0.8626501 0.8834928
## NM 0.7772999 0.8397396 0.8318980 0.8323663 0.7768719 0.7998548 0.7236015
## NV 0.8672512 0.8545221 0.8346570 0.8957633 0.8773893 0.8714882 0.8392676
## NY 0.8802316 0.8577175 0.8413395 0.8859795 0.8805587 0.8679918 0.8756449
## OH 0.8940024 0.8624864 0.8511882 0.8877660 0.8897722 0.8653512 0.8618674
## OK 0.8871868 0.8410539 0.8387121 0.8701087 0.8557139 0.8681456 0.8449436
## OR 0.9021303 0.8646987 0.8495586 0.8939174 0.8849003 0.8669488 0.8485379
## PA 0.9049058 0.8672652 0.8624913 0.8858070 0.8917801 0.8699165 0.8873575
## PR 0.4770536 0.4707951 0.5196310 0.4176452 0.4853376 0.4544563 0.4809619
## RI 0.8607922 0.7884353 0.7626527 0.7762853 0.8485468 0.7466068 0.8752843
## SC 0.8849003 0.8694550 0.8545461 0.8855942 0.8778068 0.8823670 0.8731007
## SD 0.7489054 0.7710863 0.7769190 0.7474588 0.7765357 0.7078888 0.7716023
## TN 0.9003106 0.8652520 0.8435330 0.8775704 0.8695213 0.8661915 0.8450384
## TX 0.8800368 0.8621945 0.8510308 0.8883176 0.8816090 0.8809046 0.8745295
## UT 0.8884955 0.8546768 0.8329007 0.8850268 0.8680597 0.8697360 0.8325672
## VA 0.9068284 0.8691453 0.8458409 0.8924782 0.8911072 0.8737229 0.8587969
## VI 0.4842680 0.4686653 0.4825968 0.4590773 0.4746245 0.4629879 0.4873990
## VT 0.8410220 0.8037274 0.8203757 0.8155906 0.8215102 0.8338508 0.8142307
## WA 0.9027332 0.8653036 0.8465407 0.8987049 0.8958591 0.8769306 0.8880246
## WI 0.9029317 0.8665593 0.8503380 0.8977460 0.9012593 0.8750462 0.8907390
## WV 0.6977520 0.7616678 0.7793160 0.7627380 0.6883818 0.7347956 0.6747221
## WY 0.8241356 0.7030715 0.7499268 0.7894632 0.7443945 0.7719345 0.7210167
##           DC        DE        FL        GA        GU        HI        IA
## AK 0.7967421 0.8099040 0.8132053 0.7968116 0.2022920 0.8193186 0.8073159
## AL 0.9042640 0.8757240 0.8700415 0.8620236 0.3074240 0.8492514 0.8621414
## AR 0.8351997 0.8375505 0.6997853 0.8466756 0.7468859 0.7461308 0.8516555
## AZ 0.9100360 0.8771051 0.8841111 0.8714685 0.4524165 0.8788644 0.8828577
## CA 0.9175773 0.8854521 0.8824945 0.8783043 0.6073742 0.8859390 0.8766436
## CO 0.9070178 0.8885699 0.8822342 0.8778839 0.5974091 0.8803764 0.8881395
## CT 0.9265757 0.8959044 0.8685504 0.8764348 0.5739910 0.8885509 0.8766201
## DC 0.9429775 0.8021948 0.7571735 0.7334659 0.3452915 0.6901488 0.8342459
## DE 0.9275630 0.9227164 0.8442179 0.8506121 0.3388142 0.8632638 0.8571624
## FL 0.8905654 0.8670463 0.8896294 0.8731538 0.6307922 0.8704972 0.8791502
## GA 0.9101429 0.8838374 0.8840389 0.8873335 0.6970603 0.8802701 0.8836197
## GU 0.4906613 0.5360089 0.5194292 0.5094382 1.0000000 0.5496450 0.5044908
## HI 0.9188319 0.8760880 0.8638323 0.8650845 0.5645242 0.9086958 0.8857920
## IA 0.9149164 0.8876665 0.8698279 0.8620850 0.8490284 0.8625814 0.8949620
## ID 0.8891983 0.8608178 0.8702035 0.8461064 0.4967613 0.8573488 0.8810313
## IL 0.9127167 0.8802822 0.8859934 0.8770138 0.7090184 0.8747798 0.8838805
## IN 0.9205449 0.8840366 0.8780156 0.8714087 0.4404584 0.8762161 0.8881547
## KS 0.8359029 0.8265291 0.6931672 0.8429488 0.6078724 0.7597121 0.8453841
## KY 0.9223142 0.8893274 0.8732464 0.8736503 0.5127055 0.8879025 0.8851142
## LA 0.8983261 0.8917872 0.8767150 0.8782792 0.6492277 0.8658044 0.8830461
## MA 0.9254618 0.8467616 0.7929872 0.7708800 0.5186846 0.7520584 0.8792064
## MD 0.9276024 0.8996873 0.8775492 0.8797975 0.5844544 0.8914036 0.8810113
## ME 0.9253549 0.8452039 0.7656166 0.7524666 0.6322870 0.7356526 0.8623170
## MI 0.9161315 0.8780298 0.8735119 0.8712171 0.5670154 0.8652370 0.8749190
## MN 0.9149895 0.8844349 0.8825487 0.8717570 0.4862980 0.8624488 0.8862961
## MO 0.9261875 0.8836300 0.8775036 0.8782080 0.4474340 0.8820141 0.8870815
## MS 0.8580542 0.8607491 0.8613079 0.8659421 0.5281515 0.8295886 0.8851514
## MT 0.8962980 0.8730792 0.8508071 0.8503987 0.6347783 0.8588108 0.8643196
## NC 0.9136674 0.8926017 0.8806453 0.8749484 0.5904335 0.8795988 0.8873898
## ND 0.7405888 0.6864637 0.6229522 0.5975862 0.1412556 0.6705699 0.6739248
## NE 0.8401109 0.7819883 0.6785912 0.7997089 0.9197808 0.7289692 0.8384100
## NH 0.9258837 0.8842085 0.8587791 0.8657812 0.6003986 0.8769073 0.8812447
## NJ 0.9217573 0.8843614 0.8787203 0.8757287 0.7080219 0.8783960 0.8787342
## NM 0.8334163 0.8309391 0.6934550 0.8254429 0.2855007 0.7330111 0.8264741
## NV 0.8960673 0.8660138 0.8855840 0.8728877 0.5700050 0.8688158 0.8842113
## NY 0.9032401 0.8769901 0.8820024 0.8709759 0.6751370 0.8689497 0.8810450
## OH 0.9237825 0.8887773 0.8725152 0.8790300 0.6233184 0.8800745 0.8869435
## OK 0.9104439 0.8817937 0.8573111 0.8589408 0.6243149 0.8693521 0.8773987
## OR 0.9210372 0.8790315 0.8722516 0.8672799 0.4833084 0.8723257 0.8874290
## PA 0.9294533 0.8963750 0.8724319 0.8746823 0.5460887 0.8782800 0.8845705
## PR 0.4693215 0.4568706 0.4637137 0.4918110 0.3447932 0.4691927 0.4235761
## RI 0.9229527 0.8351700 0.7919121 0.7647062 0.6362730 0.7455847 0.8603165
## SC 0.8849424 0.8918513 0.8821297 0.8804433 0.5266567 0.8653680 0.8874260
## SD 0.8371546 0.7833576 0.6869949 0.7378906 0.8161435 0.6804977 0.8279740
## TN 0.9254421 0.8886363 0.8733284 0.8732045 0.3562531 0.8690162 0.8813759
## TX 0.8940646 0.8799182 0.8841717 0.8754536 0.7638266 0.8727615 0.8803531
## UT 0.8907763 0.8578352 0.8788203 0.8663191 0.4514200 0.8647923 0.8822396
## VA 0.9208262 0.8765171 0.8825551 0.8761631 0.4887892 0.8862118 0.8833999
## VI 0.4562135 0.4731312 0.4358397 0.4446683 0.4711011 0.5249049 0.4182208
## VT 0.8782366 0.8484545 0.7998158 0.8177041 0.7872446 0.8529118 0.8546001
## WA 0.9180161 0.8908258 0.8806139 0.8727884 0.5565521 0.8817477 0.8814899
## WI 0.9223395 0.9032200 0.8801775 0.8787024 0.6163428 0.8834213 0.8850643
## WV 0.7874794 0.7883277 0.6421078 0.7531701 0.4005979 0.7409880 0.7945666
## WY 0.8023566 0.7696914 0.7714204 0.7199657 0.3298455 0.7899804 0.8408153
##           ID        IL        IN        KS        KY        LA        MA
## AK 0.7736650 0.7849379 0.8219164 0.7866851 0.7816101 0.7820224 0.7829593
## AL 0.8531254 0.7831553 0.8860037 0.8788698 0.8809159 0.8346056 0.8241414
## AR 0.8360959 0.7493158 0.8655787 0.8838767 0.8562229 0.8229287 0.7659127
## AZ 0.8625649 0.8957912 0.8894899 0.8898370 0.8880635 0.8433306 0.8850297
## CA 0.8532249 0.9031425 0.8882975 0.8874238 0.8930116 0.8486412 0.8936436
## CO 0.8687955 0.9025569 0.8930270 0.8935190 0.8903287 0.8684527 0.8911359
## CT 0.8407220 0.9025161 0.8923379 0.8951419 0.8970503 0.8560957 0.9060078
## DC 0.6605659 0.7503682 0.8258493 0.6786493 0.8327821 0.7577991 0.8120644
## DE 0.8339404 0.7838838 0.8822165 0.8762760 0.8846839 0.8190753 0.8411649
## FL 0.8613994 0.8997911 0.8873880 0.8856215 0.8851519 0.8476501 0.8736458
## GA 0.8557842 0.8983333 0.8930729 0.8928435 0.8944351 0.8616746 0.8799175
## GU 0.5171955 0.5387003 0.5021417 0.5068893 0.5084218 0.5163192 0.5425299
## HI 0.8655402 0.8798193 0.8868060 0.8855403 0.8856890 0.8342796 0.8762789
## IA 0.8512644 0.8878781 0.8860472 0.8696394 0.8848129 0.8406006 0.8708636
## ID 0.8825767 0.8607591 0.8790651 0.8745354 0.8717897 0.8319572 0.8457632
## IL 0.8583319 0.9087088 0.8950852 0.8863478 0.8939219 0.8509566 0.8903971
## IN 0.8572573 0.8907819 0.9014224 0.8839575 0.8951606 0.8484054 0.8836339
## KS 0.8391982 0.7469819 0.8609404 0.9028349 0.8534483 0.8203648 0.7624325
## KY 0.8502177 0.8924181 0.8970533 0.8928565 0.9064162 0.8482021 0.8872229
## LA 0.8628440 0.8882236 0.8883160 0.8884945 0.8849864 0.8734412 0.8740267
## MA 0.6630063 0.8630445 0.8449529 0.7212283 0.8742890 0.8242925 0.9150888
## MD 0.8529975 0.9040981 0.8954098 0.8872413 0.8947280 0.8507174 0.8970530
## ME 0.6499655 0.8399338 0.8208734 0.7065720 0.8575566 0.7957144 0.8890079
## MI 0.8410247 0.9034799 0.8933744 0.8746198 0.8897412 0.8389044 0.8921840
## MN 0.8695152 0.9007261 0.8945907 0.8924827 0.8848501 0.8589412 0.8917337
## MO 0.8501627 0.8938993 0.8988529 0.8876443 0.8941623 0.8582902 0.8892354
## MS 0.8516480 0.8625532 0.8892708 0.8836165 0.8864339 0.8383120 0.8474531
## MT 0.8558770 0.8682315 0.8657123 0.8807506 0.8766321 0.8396306 0.8487258
## NC 0.8683605 0.9002396 0.8915580 0.8979850 0.8906034 0.8577505 0.8929794
## ND 0.6058725 0.6150008 0.6791784 0.5802262 0.6880002 0.5677671 0.6141798
## NE 0.8147275 0.7461747 0.8417187 0.8490454 0.8310914 0.7995985 0.7521730
## NH 0.8390916 0.8839715 0.8924784 0.8769018 0.8931229 0.8402421 0.8843438
## NJ 0.8439394 0.9061494 0.8925075 0.8786174 0.8916569 0.8490577 0.8952780
## NM 0.8444046 0.6591153 0.8536705 0.8750443 0.8473435 0.7948100 0.7099155
## NV 0.8655529 0.8839102 0.8824215 0.8889053 0.8828255 0.8469573 0.8638038
## NY 0.8567130 0.9041883 0.8889573 0.8845302 0.8849309 0.8533069 0.8865611
## OH 0.8470179 0.8963803 0.9020704 0.8880735 0.8982926 0.8531381 0.8922443
## OK 0.8571053 0.8838505 0.8788581 0.8857052 0.8858112 0.8486033 0.8737955
## OR 0.8756373 0.8847266 0.8928580 0.8886612 0.8865707 0.8579577 0.8782759
## PA 0.8665526 0.9008118 0.8987036 0.8907499 0.8951333 0.8579620 0.9009976
## PR 0.4358825 0.3782315 0.3721073 0.4508931 0.4838263 0.4924332 0.4366675
## RI 0.6620475 0.8642285 0.8387724 0.7093037 0.8635687 0.8147937 0.8906460
## SC 0.8639401 0.8934286 0.8941115 0.8910370 0.8871549 0.8613609 0.8802839
## SD 0.6580471 0.7398658 0.8066118 0.6944558 0.8186588 0.7876560 0.7476200
## TN 0.8547933 0.8691058 0.8939729 0.8879109 0.8908907 0.8470133 0.8749829
## TX 0.8694425 0.9016460 0.8905501 0.8921283 0.8893418 0.8621751 0.8836410
## UT 0.8714433 0.8791234 0.8809282 0.8947811 0.8844987 0.8412917 0.8550155
## VA 0.8597954 0.8935319 0.8980553 0.8948211 0.8980494 0.8515492 0.8809338
## VI 0.4750502 0.4573855 0.4452750 0.4743230 0.4610907 0.4835789 0.4871123
## VT 0.8215864 0.8504119 0.8425661 0.8437830 0.8618513 0.8126210 0.8512745
## WA 0.8701903 0.9017135 0.8931027 0.8982261 0.8939585 0.8576143 0.8927785
## WI 0.8549754 0.9048670 0.9012948 0.8907054 0.8994059 0.8585834 0.9025747
## WV 0.8229843 0.6133013 0.7858580 0.7929469 0.7935453 0.7034033 0.6905066
## WY 0.7572188 0.6989131 0.6956094 0.7949157 0.7977649 0.7430979 0.7435787
##           MD        ME        MI        MN        MO        MS        MT
## AK 0.8195652 0.7144713 0.7307346 0.8446923 0.7993269 0.7911838 0.8389985
## AL 0.8759002 0.8424466 0.8858892 0.8848809 0.8648646 0.8607906 0.8536073
## AR 0.8430289 0.8662716 0.8751651 0.8498882 0.8484332 0.7837079 0.8580781
## AZ 0.8869909 0.8769390 0.9050986 0.9000379 0.8745784 0.8496818 0.8663072
## CA 0.8966120 0.8744254 0.9097884 0.8974127 0.8731774 0.8498889 0.8606623
## CO 0.8908577 0.8791354 0.9025432 0.9004567 0.8773959 0.8628353 0.8777412
## CT 0.8915161 0.8859823 0.9074710 0.8925998 0.8779044 0.8534982 0.8654180
## DC 0.7862115 0.8082579 0.8576873 0.8292348 0.7948620 0.7518535 0.7647003
## DE 0.8819600 0.8528791 0.8721956 0.8750068 0.8608111 0.8316112 0.8487521
## FL 0.8804033 0.8689748 0.9092283 0.8944056 0.8676271 0.8550101 0.8662879
## GA 0.8978689 0.8754043 0.9135144 0.8951981 0.8786516 0.8673135 0.8716294
## GU 0.5115005 0.5136500 0.5026150 0.5075430 0.5073237 0.5004877 0.5038224
## HI 0.8923865 0.8731479 0.8902327 0.8832265 0.8688691 0.8388177 0.8765826
## IA 0.8867682 0.8755044 0.8912828 0.8869251 0.8703510 0.8419498 0.8637846
## ID 0.8624717 0.8570488 0.8821943 0.8877901 0.8505574 0.8461579 0.8785887
## IL 0.8861668 0.8817262 0.9161257 0.8971858 0.8768386 0.8611224 0.8673824
## IN 0.8869516 0.8796405 0.9111425 0.8907784 0.8796216 0.8616607 0.8606430
## KS 0.8454880 0.8565348 0.8618217 0.8564272 0.8442938 0.7855450 0.8583667
## KY 0.8912044 0.8836205 0.9072904 0.8882238 0.8775164 0.8588616 0.8661073
## LA 0.8924615 0.8654657 0.8949856 0.8913355 0.8695381 0.8582452 0.8737343
## MA 0.8115353 0.8907206 0.9009260 0.8588494 0.8249166 0.8201658 0.8251956
## MD 0.9056063 0.8854613 0.9082006 0.8947570 0.8823072 0.8574653 0.8676402
## ME 0.8053431 0.8966074 0.8673240 0.8306522 0.8056480 0.7668344 0.8215777
## MI 0.8823145 0.8815702 0.9165493 0.8870104 0.8745362 0.8529696 0.8465537
## MN 0.8872922 0.8779834 0.9107313 0.9053090 0.8758357 0.8577523 0.8715545
## MO 0.8938046 0.8837636 0.9134795 0.8935777 0.8872750 0.8674073 0.8667873
## MS 0.8699081 0.8617994 0.8857298 0.8793038 0.8696398 0.8818414 0.8572359
## MT 0.8689593 0.8638916 0.8680238 0.8696976 0.8525629 0.8261613 0.8985274
## NC 0.8898082 0.8819858 0.9002329 0.8988043 0.8748839 0.8589360 0.8824829
## ND 0.6539781 0.6768347 0.6765360 0.6908616 0.6334353 0.5381584 0.7208415
## NE 0.8298330 0.8247691 0.8521898 0.8437721 0.8147632 0.7466470 0.8240254
## NH 0.8877366 0.8814878 0.8988399 0.8856039 0.8783190 0.8477310 0.8714410
## NJ 0.8898913 0.8845995 0.9143162 0.8899923 0.8776713 0.8556434 0.8576551
## NM 0.8313639 0.8345589 0.8429149 0.8508599 0.8343763 0.7711490 0.8631347
## NV 0.8861353 0.8675902 0.8937860 0.8949334 0.8639220 0.8475832 0.8719937
## NY 0.8802485 0.8753737 0.9109819 0.8935228 0.8672798 0.8479132 0.8672975
## OH 0.8984407 0.8847826 0.9150578 0.8950629 0.8856179 0.8611388 0.8626376
## OK 0.8751140 0.8761049 0.8848609 0.8826408 0.8610746 0.8330886 0.8718995
## OR 0.8864679 0.8761019 0.9019296 0.8979930 0.8734869 0.8517126 0.8816940
## PA 0.8931382 0.8883035 0.9100261 0.8978476 0.8774788 0.8571632 0.8755945
## PR 0.4807375 0.4421461 0.3760710 0.4452365 0.4244541 0.5228391 0.4932467
## RI 0.8095017 0.8809020 0.8959852 0.8503938 0.8173606 0.7938346 0.8256402
## SC 0.8929470 0.8708774 0.9079471 0.8947423 0.8758560 0.8709397 0.8678973
## SD 0.7762095 0.8217310 0.8413799 0.8129374 0.7833068 0.7151722 0.8012770
## TN 0.8881613 0.8724474 0.9012914 0.8911577 0.8769145 0.8633246 0.8648097
## TX 0.8843438 0.8685568 0.9066505 0.8977920 0.8688469 0.8610449 0.8720392
## UT 0.8757105 0.8628213 0.8931326 0.8938716 0.8650098 0.8582379 0.8768458
## VA 0.8915219 0.8745160 0.9082193 0.8964510 0.8791104 0.8625847 0.8711091
## VI 0.4629319 0.4837363 0.4422091 0.4359851 0.4526495 0.4743345 0.5385273
## VT 0.8484699 0.8383729 0.8595454 0.8303242 0.8249915 0.7969208 0.8362035
## WA 0.8903569 0.8804741 0.9048034 0.9026800 0.8740530 0.8543344 0.8843076
## WI 0.9018592 0.8866976 0.9117541 0.9005158 0.8841073 0.8599006 0.8745016
## WV 0.7783782 0.8108310 0.7478834 0.7759846 0.7782298 0.7439453 0.8138673
## WY 0.7613052 0.8251123 0.6700642 0.7823369 0.6960666 0.6330846 0.8204257
##           NC        ND        NE        NH        NJ        NM        NV
## AK 0.8457522 0.9097027 0.8254549 0.7706904 0.8021818 0.7476427 0.7750186
## AL 0.8834165 0.8858058 0.8688587 0.8691180 0.6975159 0.8941284 0.8588809
## AR 0.8611631 0.9060318 0.9201709 0.8660674 0.8269649 0.8919973 0.7938609
## AZ 0.8892240 0.9103145 0.8781608 0.8921663 0.8841716 0.8891132 0.8797697
## CA 0.8872731 0.8974663 0.8821485 0.8967418 0.8983771 0.8798916 0.8778152
## CO 0.8956136 0.8891528 0.9077931 0.8951456 0.8932225 0.9007092 0.8722806
## CT 0.8924125 0.9068956 0.8985605 0.9061196 0.9026184 0.8755497 0.8575901
## DC 0.7323660 0.8804434 0.7588977 0.8300481 0.6939218 0.8100441 0.7088402
## DE 0.8797774 0.9044843 0.8763189 0.8736841 0.7213357 0.8795766 0.8363228
## FL 0.8848672 0.8873174 0.8884366 0.8830431 0.8885161 0.8841096 0.8761945
## GA 0.8923693 0.8811992 0.9003715 0.8948641 0.8748194 0.8889966 0.8720391
## GU 0.5072949 0.4995681 0.5147028 0.5123393 0.5691395 0.5068049 0.5177863
## HI 0.8863382 0.9112143 0.8909998 0.9012751 0.8557648 0.8824067 0.8533938
## IA 0.8753761 0.8769164 0.8957810 0.8922492 0.8625815 0.8879958 0.8608565
## ID 0.8711682 0.9207515 0.8929138 0.8846213 0.8190645 0.8991693 0.8603029
## IL 0.8875267 0.8976103 0.8954512 0.8921488 0.9004027 0.8860688 0.8715056
## IN 0.8853090 0.8968545 0.9039006 0.9000148 0.8662591 0.8818719 0.8493578
## KS 0.8627021 0.9053480 0.8952297 0.8590003 0.8172970 0.8882506 0.7822499
## KY 0.8885420 0.9130497 0.8985270 0.9050270 0.8723003 0.8819559 0.8544212
## LA 0.8895486 0.8636364 0.9127056 0.8953420 0.8625855 0.9015635 0.8654597
## MA 0.7656549 0.8901245 0.7884140 0.9039841 0.8782562 0.8267573 0.7236128
## MD 0.8890169 0.8917081 0.9007888 0.9001941 0.9017762 0.8835873 0.8633505
## ME 0.7542921 0.9020370 0.7788697 0.8935060 0.8631172 0.8238258 0.7097710
## MI 0.8745906 0.8972864 0.8856107 0.8861560 0.9015521 0.8681165 0.8518850
## MN 0.8879822 0.9058519 0.9032541 0.8944170 0.8905417 0.8938140 0.8728624
## MO 0.8857662 0.9010293 0.9027131 0.9034694 0.8711730 0.8809221 0.8602081
## MS 0.8785894 0.8790038 0.8827874 0.8820953 0.8359394 0.8825309 0.8361519
## MT 0.8798375 0.9109624 0.8986842 0.8891157 0.8494900 0.9036629 0.8528415
## NC 0.9004310 0.9027928 0.9062242 0.9020095 0.8924896 0.9027396 0.8696559
## ND 0.6607252 0.9675736 0.6960293 0.7085372 0.6142436 0.6747384 0.6420619
## NE 0.8146057 0.9266537 0.9232210 0.8417056 0.8154390 0.8786481 0.7757886
## NH 0.8814909 0.9270496 0.8976202 0.9104947 0.8683292 0.8832851 0.8531807
## NJ 0.8831631 0.8939394 0.8995085 0.8949389 0.9059109 0.8738514 0.8644302
## NM 0.8588508 0.9151011 0.8719577 0.8330072 0.6688194 0.9142601 0.8023594
## NV 0.8868892 0.8986180 0.8889544 0.8881055 0.8542050 0.8942691 0.8856575
## NY 0.8866528 0.8813791 0.8987125 0.8867429 0.8963418 0.8858537 0.8675569
## OH 0.8870053 0.9190959 0.8961648 0.9027574 0.8768476 0.8825001 0.8621673
## OK 0.8827830 0.9115382 0.9159102 0.8969857 0.8573280 0.8941538 0.8531737
## OR 0.8874017 0.9129418 0.9005956 0.8998462 0.8604776 0.9040032 0.8721788
## PA 0.8902558 0.9092349 0.9136278 0.9048799 0.9004823 0.8953722 0.8587929
## PR 0.4714050 0.3758547 0.5194375 0.4154036 0.4694822 0.3877378 0.4560960
## RI 0.7590684 0.8966746 0.7868709 0.8913736 0.8794735 0.8255922 0.7242420
## SC 0.8898508 0.8739653 0.9002014 0.8863089 0.8829535 0.8879713 0.8610865
## SD 0.7344550 0.8990499 0.8045787 0.8310811 0.8054921 0.8430107 0.6941182
## TN 0.8853751 0.8952710 0.9075998 0.8996929 0.8331125 0.8869311 0.8553912
## TX 0.8921937 0.8964946 0.9130714 0.8889942 0.8921166 0.8942230 0.8719507
## UT 0.8838505 0.8936515 0.8908710 0.8840739 0.8430480 0.8970080 0.8697429
## VA 0.8922529 0.9077593 0.8906855 0.9028049 0.8695623 0.8912065 0.8655332
## VI 0.4740967 0.5377528 0.4447992 0.5078225 0.4583756 0.4976308 0.4280715
## VT 0.8462103 0.8432664 0.8893615 0.8665857 0.8287392 0.8528968 0.7755173
## WA 0.8962190 0.9224070 0.8999696 0.9005255 0.8959198 0.8998783 0.8765404
## WI 0.8928930 0.9255740 0.8997506 0.9055063 0.9005699 0.8905090 0.8715488
## WV 0.8045260 0.9070035 0.8020966 0.8142091 0.6151120 0.8268800 0.7365301
## WY 0.8071550 0.8783560 0.8048775 0.8258622 0.6020633 0.8338175 0.7939226
##           NY        OH        OK        OR        PA        PR        RI
## AK 0.7563219 0.8019320 0.7743104 0.8126648 0.8052729 0.5545721 0.7670110
## AL 0.6322989 0.8321516 0.8825249 0.8681281 0.8638586 0.7177218 0.7268852
## AR 0.8091046 0.8559201 0.8739799 0.8510437 0.8693960 0.6553011 0.8139102
## AZ 0.8863930 0.8829785 0.8859181 0.8781884 0.8788853 0.7085707 0.9031127
## CA 0.8942621 0.8912818 0.8793045 0.8757904 0.8848723 0.7013715 0.9147220
## CO 0.9042100 0.8870343 0.8968188 0.8793502 0.8900800 0.7095992 0.9050987
## CT 0.8902052 0.8903421 0.8894815 0.8637415 0.8908624 0.6996975 0.9144847
## DC 0.6401068 0.8036286 0.7503486 0.7810829 0.7808352 0.6638065 0.7292429
## DE 0.6542347 0.8263760 0.8807765 0.8504622 0.8677057 0.7133752 0.7657034
## FL 0.9039005 0.8791184 0.8734088 0.8699860 0.8774383 0.7089986 0.9017102
## GA 0.8599491 0.8927478 0.8824867 0.8716527 0.8852137 0.7090962 0.8627602
## GU 0.6131835 0.5072073 0.5053006 0.5040496 0.5116822 0.5219580 0.5280736
## HI 0.8222964 0.8777858 0.8721705 0.8709006 0.8794452 0.7190355 0.8403520
## IA 0.8397059 0.8776707 0.8765524 0.8632313 0.8727539 0.6998326 0.8470630
## ID 0.8118106 0.8610462 0.8730635 0.8740443 0.8637123 0.7293126 0.8152846
## IL 0.9033733 0.8873370 0.8846611 0.8723189 0.8851455 0.6934441 0.9084521
## IN 0.8333602 0.8837003 0.8812625 0.8709907 0.8875107 0.7042167 0.8521009
## KS 0.7963076 0.8533132 0.8820676 0.8464860 0.8655974 0.6595125 0.8092528
## KY 0.8341189 0.8835395 0.8849916 0.8659957 0.8882963 0.7177743 0.8587496
## LA 0.8552564 0.8851889 0.8924014 0.8761224 0.8836521 0.7064613 0.8588208
## MA 0.8810520 0.8743350 0.7851047 0.7981184 0.8223915 0.6975805 0.9124091
## MD 0.8856878 0.8875145 0.8838498 0.8710962 0.8932818 0.7017694 0.9129996
## ME 0.8509229 0.8566913 0.7749835 0.7820446 0.8038576 0.6953509 0.8946327
## MI 0.8988294 0.8863200 0.8732813 0.8626861 0.8814731 0.6901936 0.9055899
## MN 0.8971329 0.8864412 0.8902945 0.8815070 0.8889612 0.7090512 0.9044115
## MO 0.8405904 0.8917084 0.8870099 0.8705879 0.8901502 0.7168209 0.8646338
## MS 0.8213527 0.8701146 0.8784708 0.8560050 0.8700273 0.6921379 0.8230923
## MT 0.8317264 0.8638118 0.8747716 0.8649297 0.8671701 0.7189754 0.8313895
## NC 0.8923250 0.8811277 0.8904055 0.8786610 0.8904946 0.7294552 0.8981987
## ND 0.6039200 0.6687812 0.6786631 0.6840107 0.6552367 0.5185986 0.6369928
## NE 0.7955663 0.8414941 0.8531827 0.8283706 0.8432733 0.6148308 0.8214394
## NH 0.8256701 0.8843999 0.8800441 0.8638639 0.8823446 0.7157549 0.8573348
## NJ 0.9065050 0.8865730 0.8785680 0.8635472 0.8869814 0.6943750 0.9111683
## NM 0.6026710 0.7998898 0.8794551 0.8544030 0.8382138 0.6624402 0.6950918
## NV 0.8477721 0.8781097 0.8820269 0.8742337 0.8699539 0.7202742 0.8499946
## NY 0.9121879 0.8818549 0.8750935 0.8671762 0.8820801 0.7110481 0.9035881
## OH 0.8482584 0.9021625 0.8854941 0.8741525 0.8866779 0.7122342 0.8669589
## OK 0.8333371 0.8710575 0.9044016 0.8597215 0.8747669 0.7125945 0.8485833
## OR 0.8429816 0.8858957 0.8843861 0.8876234 0.8852140 0.7236748 0.8575422
## PA 0.8878544 0.8874047 0.8870485 0.8809272 0.9006620 0.7170236 0.9103765
## PR 0.4143281 0.4058465 0.3652222 0.4240687 0.4475907 0.7881299 0.3895039
## RI 0.8896967 0.8661934 0.7800693 0.7946893 0.8126417 0.6740085 0.9213953
## SC 0.8986027 0.8863589 0.8885309 0.8715129 0.8848249 0.7091713 0.8908383
## SD 0.7985583 0.8291765 0.7657554 0.7786591 0.7918227 0.6143128 0.8257027
## TN 0.7803009 0.8752483 0.8841384 0.8672967 0.8867474 0.7244330 0.8306997
## TX 0.9084935 0.8835017 0.8881459 0.8733405 0.8862507 0.7186602 0.9039071
## UT 0.8284052 0.8653545 0.8784415 0.8704341 0.8692076 0.7192682 0.8357974
## VA 0.8390540 0.8855636 0.8840528 0.8788847 0.8882804 0.7148691 0.8605994
## VI 0.4794400 0.4734288 0.4928891 0.4743845 0.4510799 0.5004317 0.4764599
## VT 0.8107880 0.8412480 0.8475131 0.8167255 0.8431748 0.6845859 0.8226758
## WA 0.9006355 0.8882120 0.8922704 0.8840068 0.8902966 0.7187352 0.9101999
## WI 0.8964427 0.8990024 0.8864985 0.8812672 0.8936589 0.7161078 0.9169540
## WV 0.5371780 0.7636490 0.7849779 0.8142985 0.7782191 0.6467506 0.6199067
## WY 0.5795905 0.6399696 0.8276918 0.7827446 0.7411963 0.7249810 0.6585061
##           SC        SD        TN        TX        UT        VA        VI
## AK 0.8187874 0.8363327 0.7858682 0.7658834 0.7988854 0.8440409 0.5918367
## AL 0.8529602 0.9145326 0.8720155 0.8626072 0.8855779 0.8939002 0.3945578
## AR 0.7659726 0.9341861 0.8424275 0.8426025 0.8525266 0.8403229 0.7052154
## AZ 0.8523793 0.9321405 0.8630126 0.8751271 0.8916388 0.8971836 0.9183673
## CA 0.8489018 0.9309407 0.8654764 0.8784689 0.8873696 0.8994442 0.9319728
## CO 0.8688392 0.9515696 0.8725972 0.8886915 0.8956314 0.8966708 0.9682540
## CT 0.8525247 0.9095868 0.8744706 0.8770788 0.8782138 0.8975412 0.9433107
## DC 0.7016551 0.8994191 0.8005679 0.6909160 0.7689553 0.7943485 0.5170068
## DE 0.8430738 0.8826312 0.8680833 0.8493057 0.8652034 0.8800555 0.6757370
## FL 0.8601879 0.9373365 0.8622501 0.8792855 0.8890795 0.8898492 0.9365079
## GA 0.8632866 0.9455708 0.8744960 0.8851409 0.8912636 0.8986196 0.6394558
## GU 0.5128689 0.4991843 0.5073473 0.5101910 0.5031185 0.5147232 0.5759637
## HI 0.8509600 0.9298100 0.8643401 0.8617682 0.8816137 0.8914556 0.6258503
## IA 0.8466997 0.9051676 0.8584270 0.8615115 0.8799966 0.8797945 0.6281179
## ID 0.8440665 0.9294130 0.8535250 0.8583433 0.8929017 0.8693079 0.6054422
## IL 0.8551260 0.9432663 0.8680157 0.8800957 0.8872492 0.8984956 0.9501134
## IN 0.8599118 0.9427484 0.8745835 0.8715502 0.8821425 0.8974091 0.5646259
## KS 0.7602799 0.9071355 0.8403990 0.8371599 0.8597017 0.8399203 0.7029478
## KY 0.8548456 0.9135831 0.8761999 0.8719912 0.8845526 0.8984747 0.6031746
## LA 0.8673141 0.9526054 0.8685127 0.8877362 0.8912096 0.8888071 0.6235828
## MA 0.7635768 0.9077138 0.8212111 0.7244997 0.8061418 0.8149268 0.9092971
## MD 0.8574633 0.9107866 0.8743293 0.8757635 0.8893810 0.8980715 0.9274376
## ME 0.7350059 0.9163365 0.8022243 0.7019649 0.7781120 0.7953267 0.8798186
## MI 0.8380379 0.9338322 0.8646870 0.8686165 0.8739828 0.8916044 0.9319728
## MN 0.8518989 0.9402626 0.8693354 0.8809665 0.8926785 0.8952934 0.9319728
## MO 0.8573275 0.9052970 0.8777418 0.8760775 0.8879078 0.8968762 0.5850340
## MS 0.8620017 0.8981503 0.8685665 0.8638838 0.8808449 0.8809459 0.6077098
## MT 0.8356495 0.9472194 0.8492574 0.8585274 0.8771016 0.8763748 0.6485261
## NC 0.8626021 0.9503353 0.8739126 0.8828519 0.8966144 0.8980796 0.9659864
## ND 0.5829928 0.7034016 0.6445208 0.5689237 0.6095317 0.6659532 0.4489796
## NE 0.7211186 0.9073858 0.8222078 0.8037179 0.8261016 0.8122400 0.7006803
## NH 0.8366725 0.9043735 0.8701040 0.8616073 0.8753418 0.8895846 0.6462585
## NJ 0.8486451 0.9040110 0.8704034 0.8767558 0.8813982 0.8934938 0.9591837
## NM 0.7624332 0.9165868 0.8347161 0.8169690 0.8591276 0.8321543 0.5192744
## NV 0.8557637 0.9306473 0.8617617 0.8766787 0.8942990 0.8853007 0.6167800
## NY 0.8531803 0.9443711 0.8648281 0.8806531 0.8826542 0.8909063 0.9523810
## OH 0.8544980 0.9370086 0.8724255 0.8727393 0.8804285 0.8999560 0.6303855
## OK 0.8438435 0.9207126 0.8628745 0.8688858 0.8802240 0.8760473 0.6349206
## OR 0.8435892 0.9423513 0.8709292 0.8765900 0.8946035 0.8953666 0.6167800
## PA 0.8483083 0.9501368 0.8790298 0.8786518 0.8893979 0.8997706 0.9365079
## PR 0.4768442 0.5105000 0.4766450 0.4542604 0.4954694 0.4455291 0.4081633
## RI 0.7490567 0.9101133 0.8116138 0.7209664 0.7989687 0.8098841 0.9433107
## SC 0.8786391 0.9371467 0.8704814 0.8841772 0.8911290 0.8893968 0.8888889
## SD 0.6739406 0.9614525 0.7844122 0.6966431 0.7657352 0.7747516 0.7641723
## TN 0.8493848 0.9060739 0.8831626 0.8727251 0.8879061 0.8934454 0.4761905
## TX 0.8688062 0.9454500 0.8683703 0.8913043 0.8928947 0.8893625 0.9410431
## UT 0.8532774 0.9024919 0.8610203 0.8721008 0.9018565 0.8796214 0.6077098
## VA 0.8536471 0.9475647 0.8739549 0.8764939 0.8921947 0.9072152 0.6213152
## VI 0.4547630 0.4517465 0.4468281 0.4467297 0.4577301 0.4774048 1.0000000
## VT 0.8140200 0.8812329 0.8097666 0.8354583 0.8368603 0.8385833 0.6326531
## WA 0.8554949 0.9441639 0.8711858 0.8843352 0.8951371 0.9005064 0.9682540
## WI 0.8615033 0.9406164 0.8722097 0.8790626 0.8877668 0.9023293 0.9387755
## WV 0.7082002 0.7698585 0.7851715 0.7120203 0.8128400 0.7719960 0.5102041
## WY 0.7281498 0.8355818 0.7265881 0.7612785 0.7826609 0.7639090 0.3560091
##           VT        WA        WI        WV        WY
## AK 0.6969166 0.8202492 0.8163922 0.7661033 0.7552531
## AL 0.7786461 0.8625202 0.8400527 0.8922944 0.8851265
## AR 0.7978736 0.8083930 0.8294668 0.9007353 0.8256737
## AZ 0.7971655 0.8790533 0.8897562 0.9015737 0.8857867
## CA 0.8063405 0.8797372 0.8926453 0.8997039 0.8739458
## CO 0.8311555 0.8807696 0.8902431 0.8970522 0.8916432
## CT 0.8360841 0.8781414 0.8889288 0.8901284 0.8678408
## DC 0.6839502 0.7709173 0.8056450 0.7871972 0.8243107
## DE 0.8142238 0.8597202 0.8388213 0.8827089 0.8515703
## FL 0.8009613 0.8730884 0.8828908 0.9023789 0.8890806
## GA 0.8277974 0.8735244 0.8887151 0.9010547 0.8834867
## GU 0.5711297 0.5078292 0.5058140 0.4986692 0.5053525
## HI 0.8366270 0.8680230 0.8826308 0.9092128 0.8766718
## IA 0.8369724 0.8573013 0.8840026 0.9041722 0.8845231
## ID 0.8017423 0.8604696 0.8724442 0.9045681 0.8851904
## IL 0.8207081 0.8784214 0.8890188 0.8973649 0.8803490
## IN 0.8414955 0.8658038 0.8865705 0.8935254 0.8770906
## KS 0.7563824 0.8151860 0.8323024 0.8861891 0.8436125
## KY 0.8474348 0.8704909 0.8850876 0.9029445 0.8776088
## LA 0.8430812 0.8674423 0.8828507 0.8940877 0.8828833
## MA 0.8163931 0.8023081 0.8875458 0.8331015 0.8519536
## MD 0.8405922 0.8761027 0.8894144 0.8974880 0.8727390
## ME 0.7919408 0.7885370 0.8776912 0.8328088 0.8498239
## MI 0.8162794 0.8687864 0.8832361 0.8880656 0.8561561
## MN 0.8161656 0.8796431 0.8909554 0.9018466 0.8764730
## MO 0.8424933 0.8695305 0.8878241 0.9025253 0.8758909
## MS 0.7917176 0.8512076 0.8781740 0.8908637 0.8991822
## MT 0.8285334 0.8672672 0.8661634 0.8945868 0.8713547
## NC 0.8247076 0.8820100 0.8874328 0.9022125 0.8949655
## ND 0.5163995 0.6824435 0.6888575 0.7207014 0.6819789
## NE 0.7109977 0.7990667 0.8193785 0.8520295 0.8199520
## NH 0.8219140 0.8679186 0.8836352 0.8995641 0.8715251
## NJ 0.8347409 0.8727040 0.8836482 0.8966363 0.8712767
## NM 0.7207596 0.8116903 0.7933097 0.8975047 0.8480350
## NV 0.7854136 0.8705280 0.8851823 0.9091729 0.8971732
## NY 0.8188714 0.8708489 0.8812473 0.8953753 0.8804626
## OH 0.8296535 0.8718252 0.8968936 0.8970622 0.8716742
## OK 0.8477502 0.8605133 0.8733420 0.8938515 0.8828620
## OR 0.8126574 0.8780714 0.8892243 0.9066975 0.8796320
## PA 0.8461667 0.8800983 0.8898740 0.9031574 0.8649090
## PR 0.4115331 0.4545504 0.4206242 0.4639140 0.4945623
## RI 0.7894453 0.8023087 0.8814623 0.8250732 0.8391331
## SC 0.8293853 0.8696095 0.8854649 0.9027848 0.8911747
## SD 0.6898680 0.7516001 0.8151438 0.7947365 0.8091263
## TN 0.8108830 0.8667063 0.8722179 0.8979106 0.8772965
## TX 0.8378243 0.8757097 0.8832104 0.8974082 0.8861558
## UT 0.7987619 0.8662670 0.8758840 0.9025286 0.8968964
## VA 0.8155691 0.8778822 0.8911092 0.8956581 0.8828762
## VI 0.5447184 0.4835076 0.4515728 0.4483780 0.5068291
## VT 0.9000708 0.8156328 0.8335154 0.8663395 0.8279240
## WA 0.8154468 0.8891400 0.8947131 0.8982832 0.8803135
## WI 0.8275700 0.8852146 0.9019703 0.9000366 0.8775378
## WV 0.6621457 0.7589849 0.7599596 0.9302369 0.8346821
## WY 0.7420813 0.7771514 0.7937100 0.8772791 0.9118537
```

let's plot some map

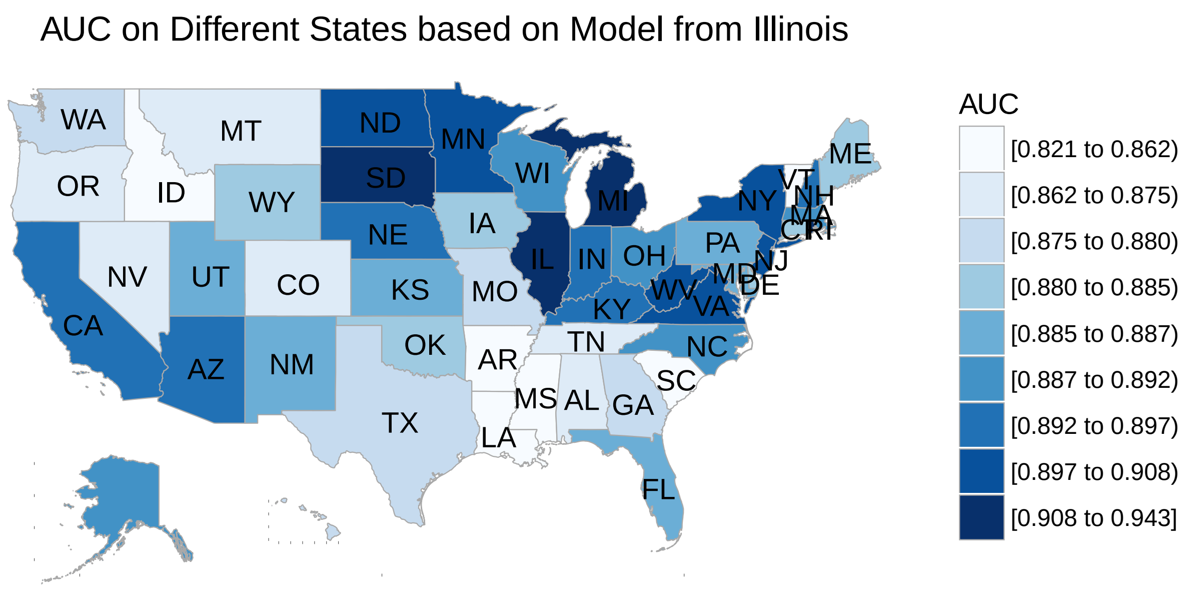
below is a fuction for transfer abberation name to full name

```
#'x' is the column of a data.frame that holds 2 digit state codes
stateFromLower <-function(x) {
  #read 52 state codes into local variable [includes DC (Washington D.C. and PR (Puerto Rico)]
  st.codes<-data.frame(
    state=as.factor(c("AK", "AL", "AR", "AZ", "CA", "CO", "CT", "DC", "DE", "FL", "GA",
                      "HI", "IA", "ID", "IL", "IN", "KS", "KY", "LA", "MA", "MD", "ME",
                      "MI", "MN", "MO", "MS",  "MT", "NC", "ND", "NE", "NH", "NJ", "NM",
                      "NV", "NY", "OH", "OK", "OR", "PA", "PR", "RI", "SC", "SD", "TN",
                      "TX", "UT", "VA", "VT", "WA", "WI", "WV", "WY")),
    full=as.factor(c("alaska","alabama","arkansas","arizona","california","colorado",
                     "connecticut","district of columbia","delaware","florida","georgia",
                     "hawaii","iowa","idaho","illinois","indiana","kansas","kentucky",
                     "louisiana","massachusetts","maryland","maine","michigan","minnesota",
                     "missouri","mississippi","montana","north carolina","north dakota",
                     "nebraska","new hampshire","new jersey","new mexico","nevada",
                     "new york","ohio","oklahoma","oregon","pennsylvania","puerto rico",
                     "rhode island","south carolina","south dakota","tennessee","texas",
                     "utah","virginia","vermont","washington","wisconsin",
                     "west virginia","wyoming"))
  )
  #create an nx1 data.frame of state codes from source column
  st.x<-data.frame(state=x)
  #match source codes with codes from 'st.codes' local variable and use to return the full state name
  refac.x<-st.codes$full[match(st.x$state,st.codes$state)]
  #return the full state names in the same order in which they appeared in the original source
  return(refac.x)
}
```

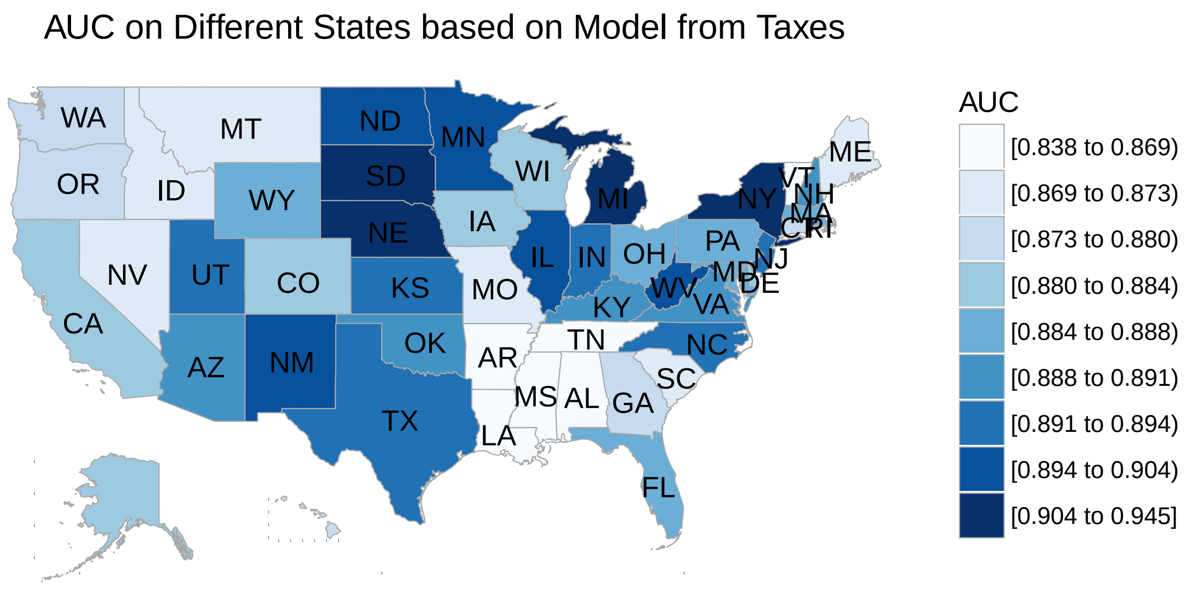
```
temp = as.data.frame(cbind(unlist(auc_matrix["CA",]),rownames(auc_matrix)))
colnames(temp)=c("value","region")
temp$region = stateFromLower(temp$region)
temp$value = as.numeric(as.character(temp$value))

temp = temp[complete.cases(temp),]


state_choropleth(temp, title = "AUC on Different States based on Model from California",num_colors = 9,l
```



```
temp = as.data.frame(cbind(unlist(auc_matrix["IL",]),rownames(auc_matrix)))
colnames(temp)=c("value","region")
temp$region = stateFromLower(temp$region)
temp$value = as.numeric(as.character(temp$value))

temp = temp[complete.cases(temp),]

state_choropleth(temp, title = "AUC on Different States based on Model from Illinois",num_colors = 9,le
```



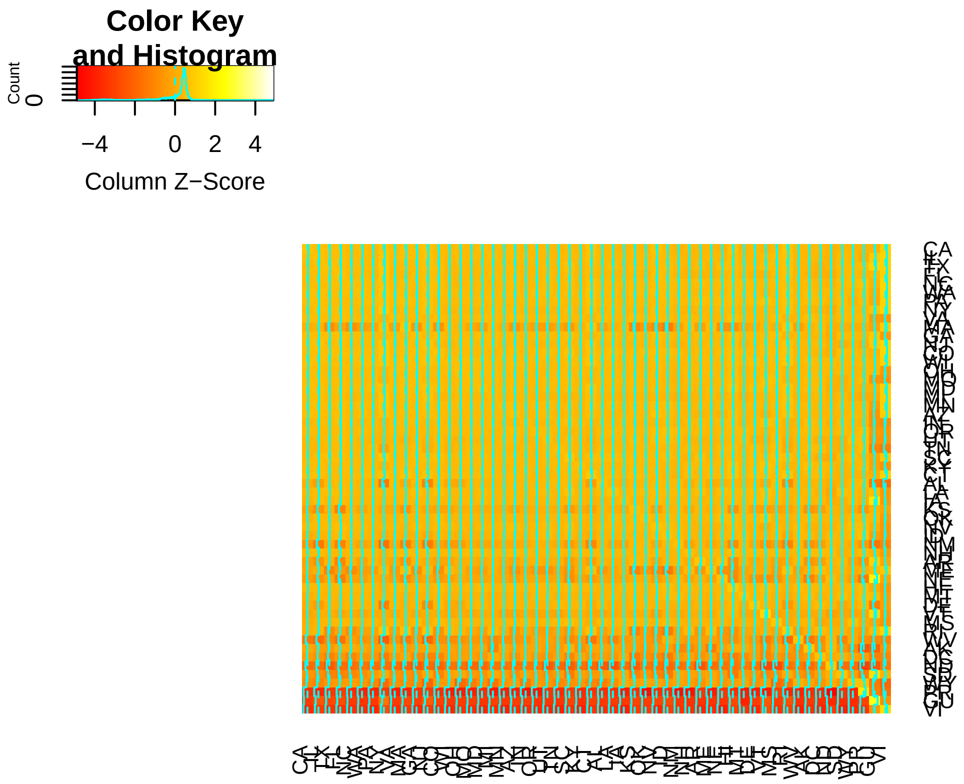
```
temp = as.data.frame(cbind(unlist(auc_matrix["TX",]),rownames(auc_matrix)))
colnames(temp)=c("value","region")
temp$region = stateFromLower(temp$region)
temp$value = as.numeric(as.character(temp$value))

temp = temp[complete.cases(temp),]

state_choropleth(temp, title = "AUC on Different States based on Model from Taxes",num_colors = 9,legend
```



```
state.order = names(sort(table(state),decreasing=T))
temp <- auc_matrix[,c(state.order)]
temp <- temp[c(state.order),]
write.csv(temp, "tempt.csv")
heatmap <- heatmap.2(temp, Rowv=F, Colv=F, col = heat.colors(256), scale="column", margins=c(5,10))
```

```
## Warning in heatmap.2(temp, Rowv = F, Colv = F, col = heat.colors(256),
## scale = "column", : Discrepancy: Rowv is FALSE, while dendrogram is `both'.
## Omitting row dendogram.
```

```
## Warning in heatmap.2(temp, Rowv = F, Colv = F, col = heat.colors(256),
## scale = "column", : Discrepancy: Colv is FALSE, while dendrogram is
## `column'. Omitting column dendogram.
```


Color Key
and Histogram
Count
Column Z−Score