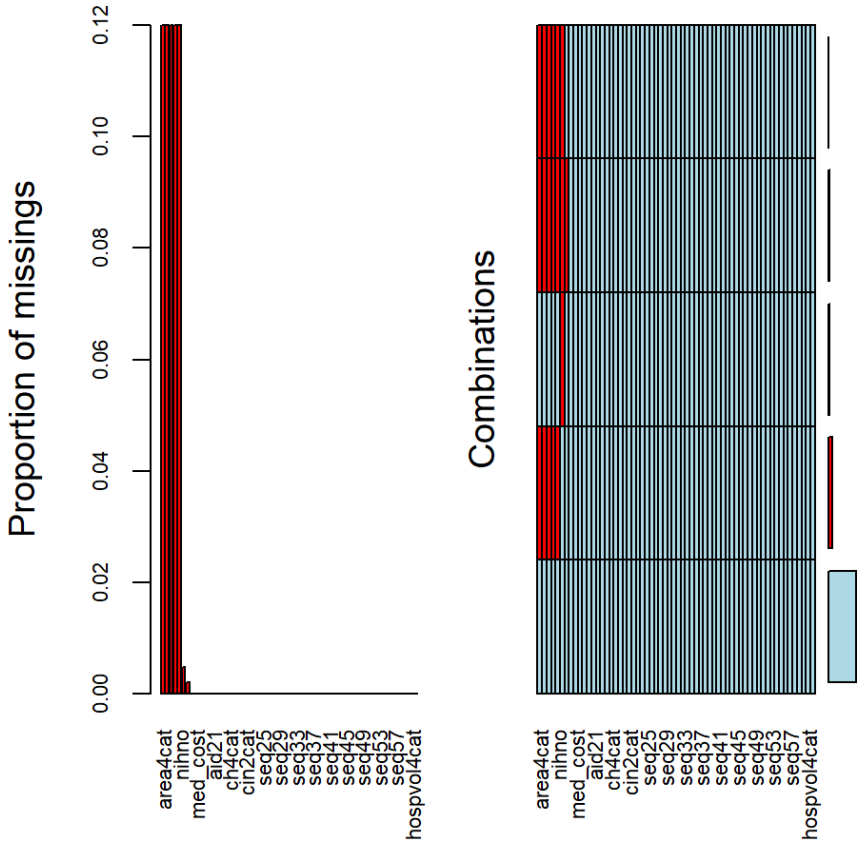
股骨頸骨折 (Femoral Neck Fracture)

710533116 許哲瑋 710633108 葉家齊

1 探索性資料分析

我們的探索性資料分析先由遺失值檢查， 再到連續型變數資料探索， 最後對類別型進行探索性資料分析。

1.1 遺失值結構檢查

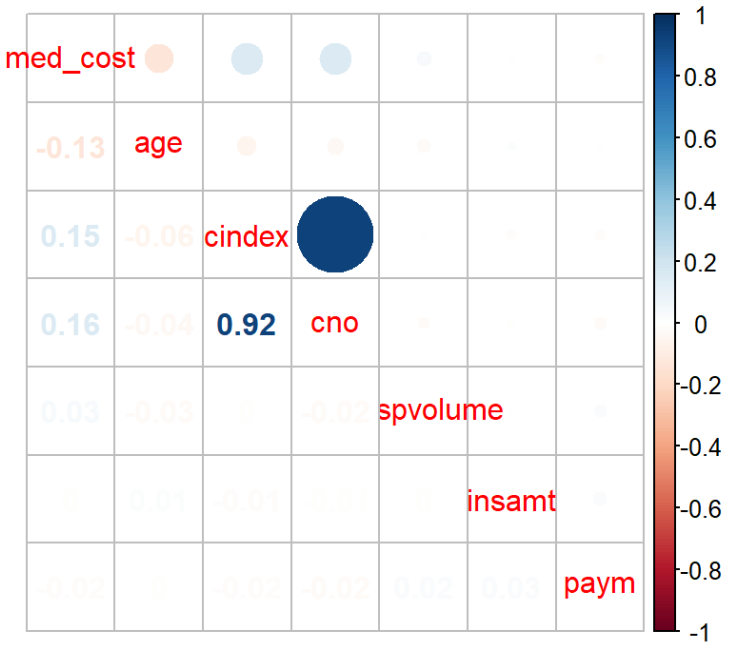


由圖可以得知， 多數的遺失值是同時遺失的， 且遺失的變數為病患居住地，其他遺失變數皆是從患者居住地分出來的， 有 3002 個遺失。 而原始保費 (insamt) 與 調整後平均保費 (paym)， 分別有 54 與 119 個遺失值。

1.2 連續型變數資料探索

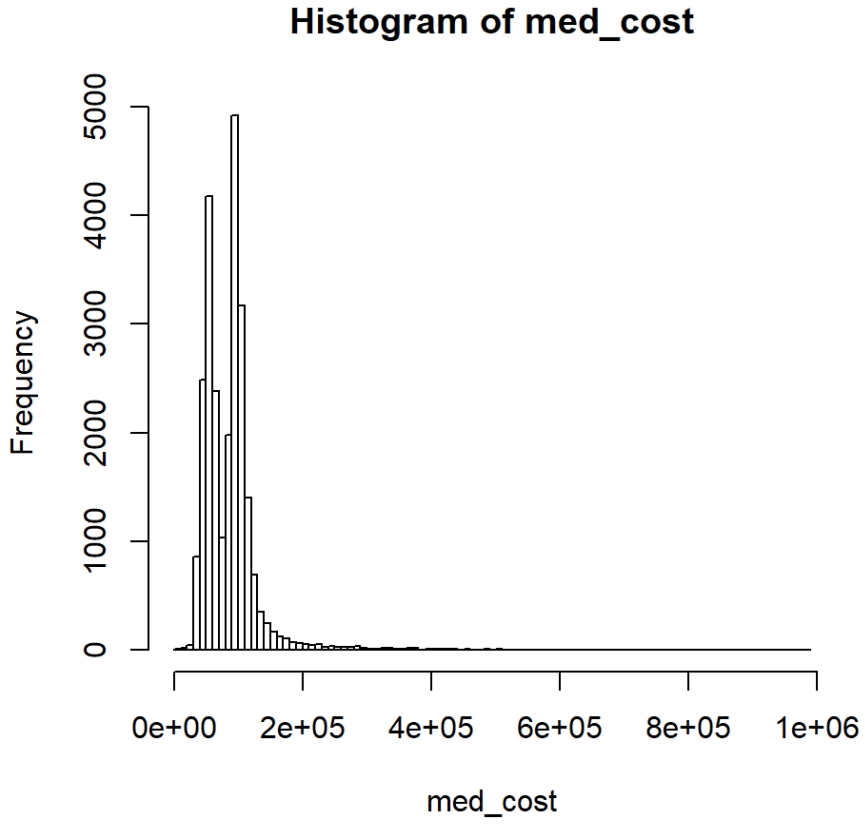
連續型變數資料探索，將由相關係數圖檢查， 最後劃出連續型變數之直方圖表進行分析。

```
library(corrplot)
cor_num <- cor( na.omit(train[,names(train)%in%c("med_cost", "age",
               "cindex", "cno", "hospvolume", "insamt", "paym")] ))
corrplot.mixed(cor_num)
```



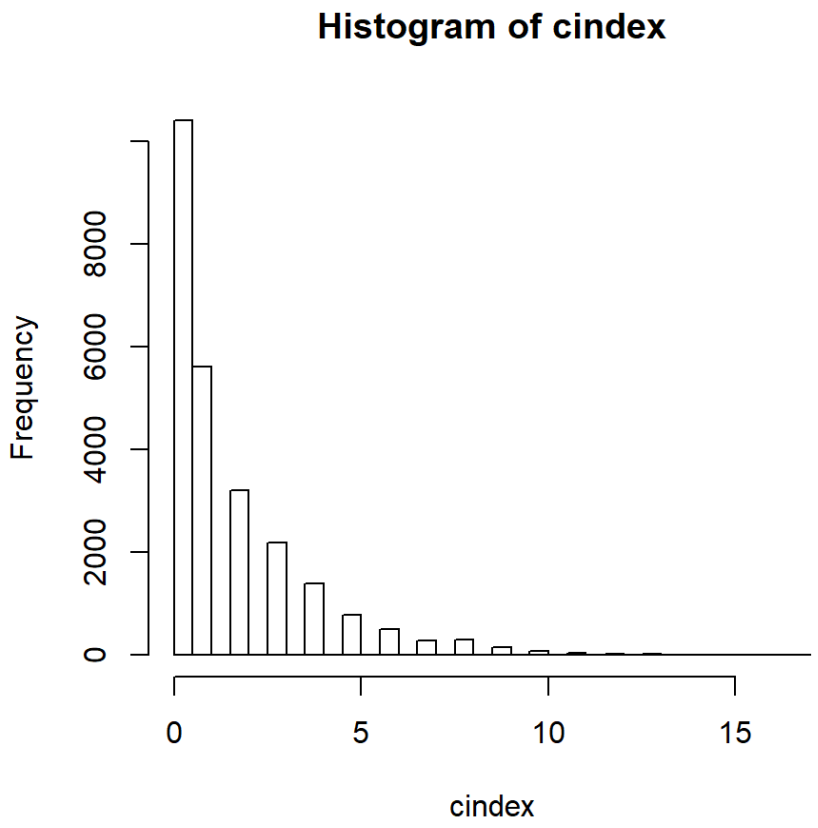
由上圖可以看出， cindex 與 cno 有高度相關， 由於 cindex 的分散程度較廣， 較能分別出個體之間的差異， 故在此選擇 cindex。 由於 aid19 與 aid21 是由 age 切割而成的變數； cin2cat、 cin3cat、 cin4cat 是由 cindex 切割而成的變數； ch2cat ch3cat ch4cat 是由 cno 切割而成的變數。

```
hist(train$med_cost,breaks=100,xlab="med_cost",main="Histogram of med_cost")
```



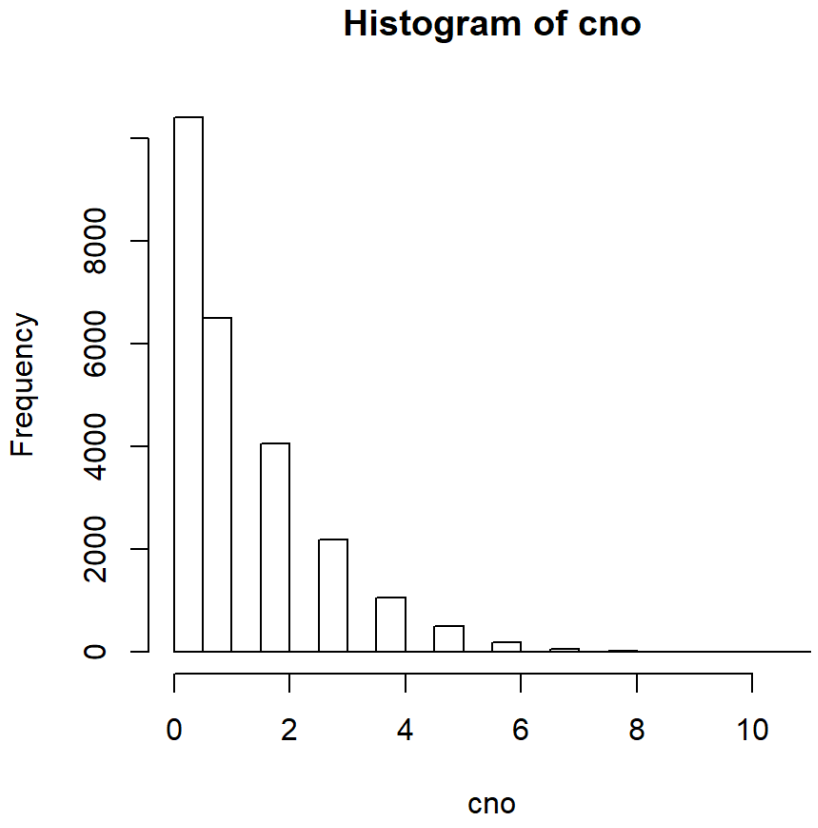
由此圖可以得知， 醫療花費大多介於 2 萬元以下， 而有少數人花費特別高， 其中有雙峰， 可能與手術型態有關係。

```
hist(train$cindex,breaks=30,xlab="cindex",main="Histogram of cindex")
```



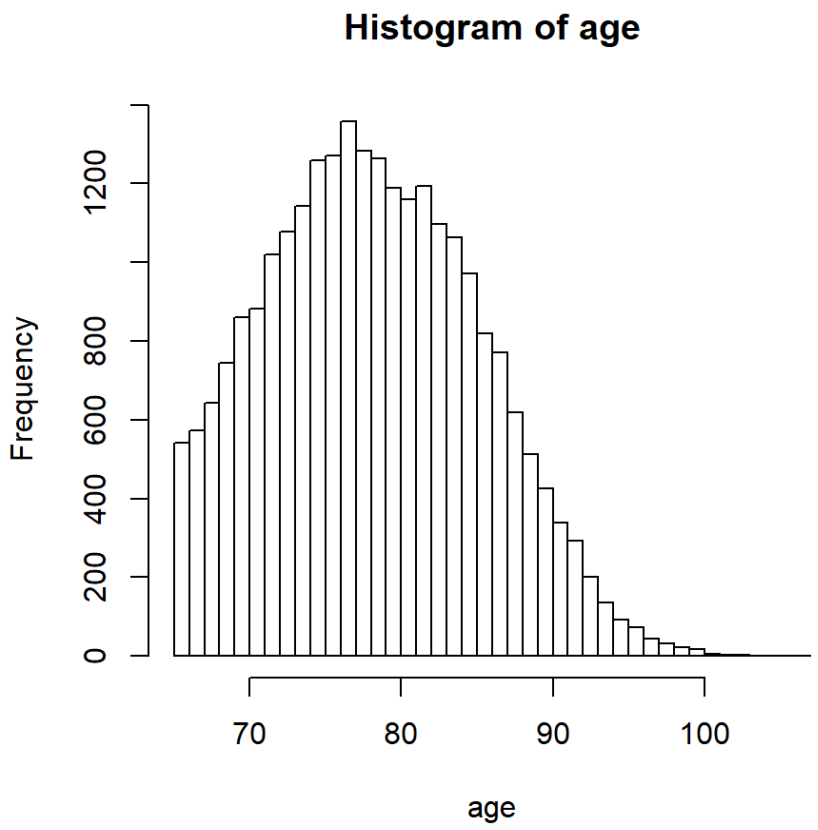
由此圖可以看到大多數人卡爾森分數都滿低的， 只有少部分的人特別高。

```
hist(train$cno,breaks=30,xlab="cno",main="Histogram of cno")
```



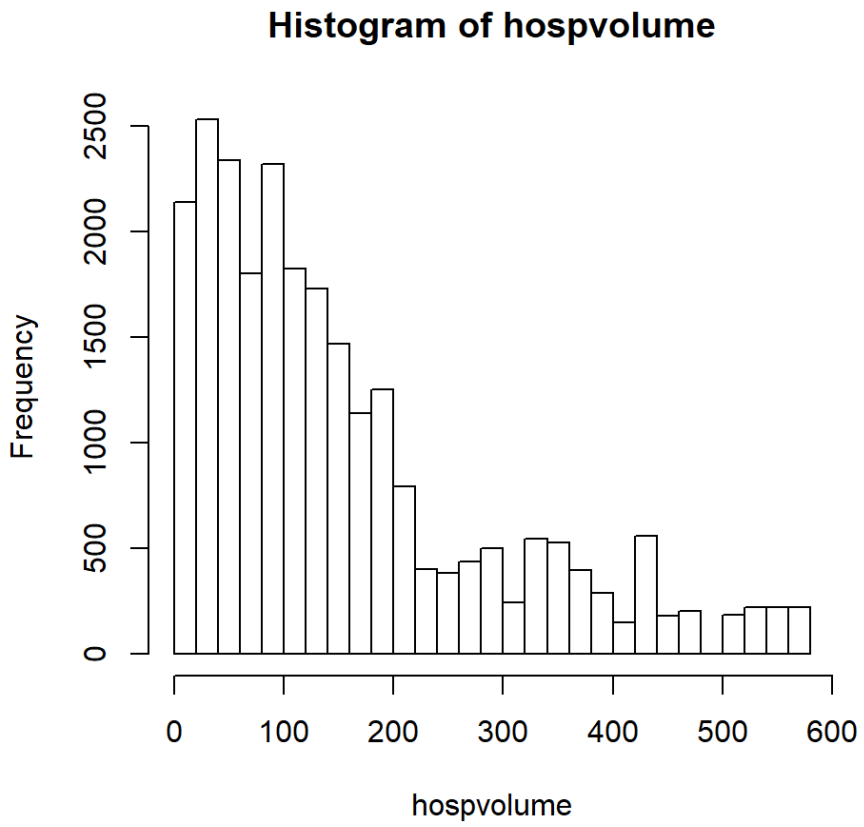
由此圖可以得知， 此 cno 變數與 cindex 變數相似程度相當高。

```
hist(train$age,breaks=30,xlab="age",main="Histogram of age")
```



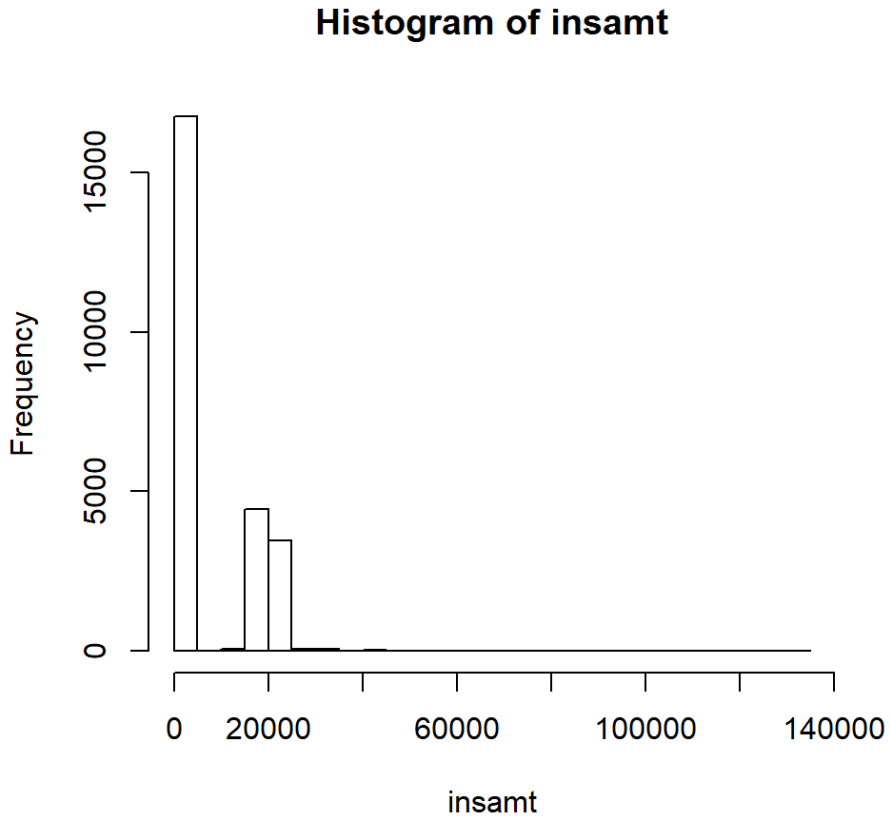
由上方的圖得知， 除了年齡最小值約為60幾歲， 故病患年齡都相當大。

```
hist(train$hospvolume,breaks=30,xlab="hospvolume",main="Histogram of hospvolume")
```



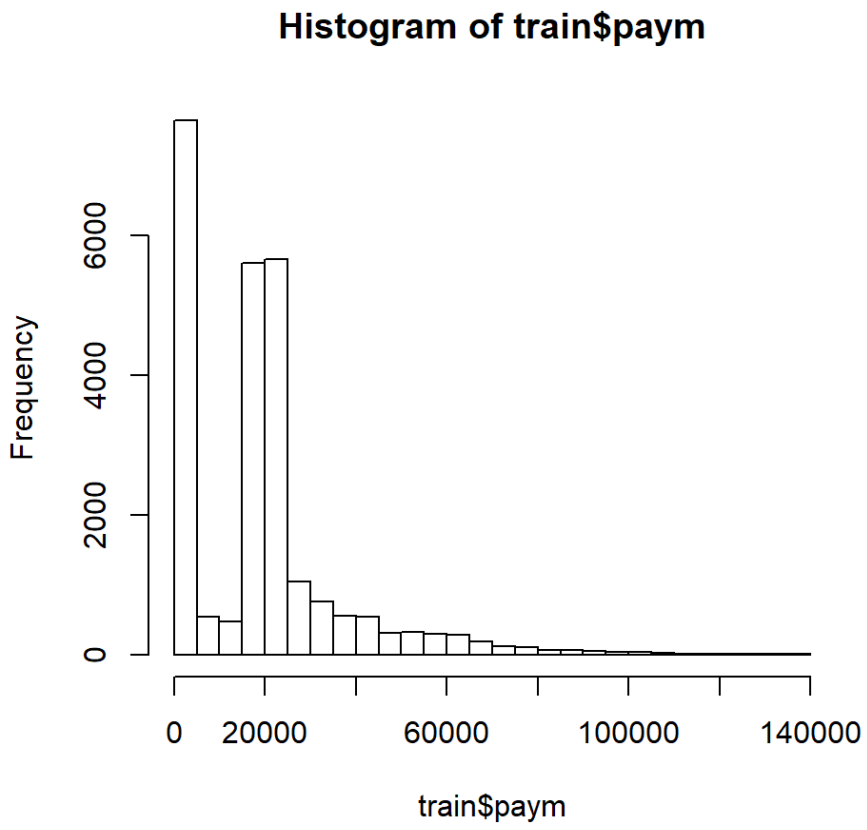
此變數呈現右邊， 在預測時須將其標準化。

```
hist(train$insamt,breaks=30,xlab="insamt",main="Histogram of insamt")
```



由圖可以得知， 大部分的人原始保費為 0元， 其次的多數介於 15000~25000 之間， 此變數之後需要做標準化或是刪除。

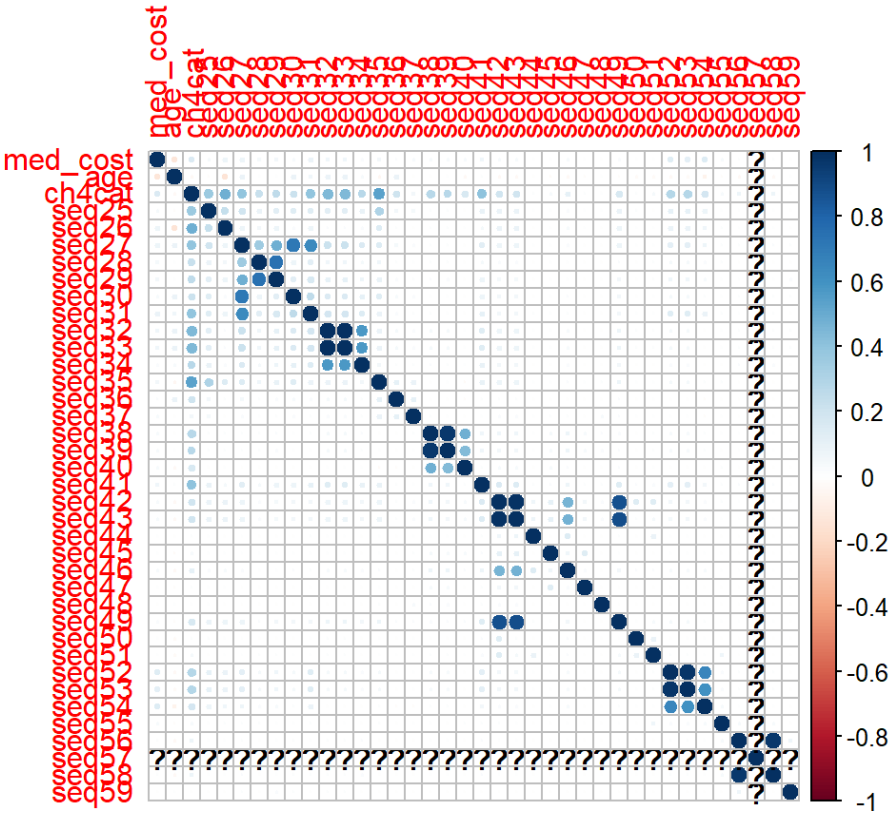
```
hist(train$paym,breaks=30)
```



由圖可知，條整後保費與原始保費大多數的都落在0 與 15000~25000 之間， 此變數之後需要做標準化。

1.3 類別型變數資料探索

```
library(corrplot)
seq_var <-train[,c(2,4,10,grep("seq",names(train) ))]
seq_cor <- cor(seq_var)
corrplot(seq_cor)
```



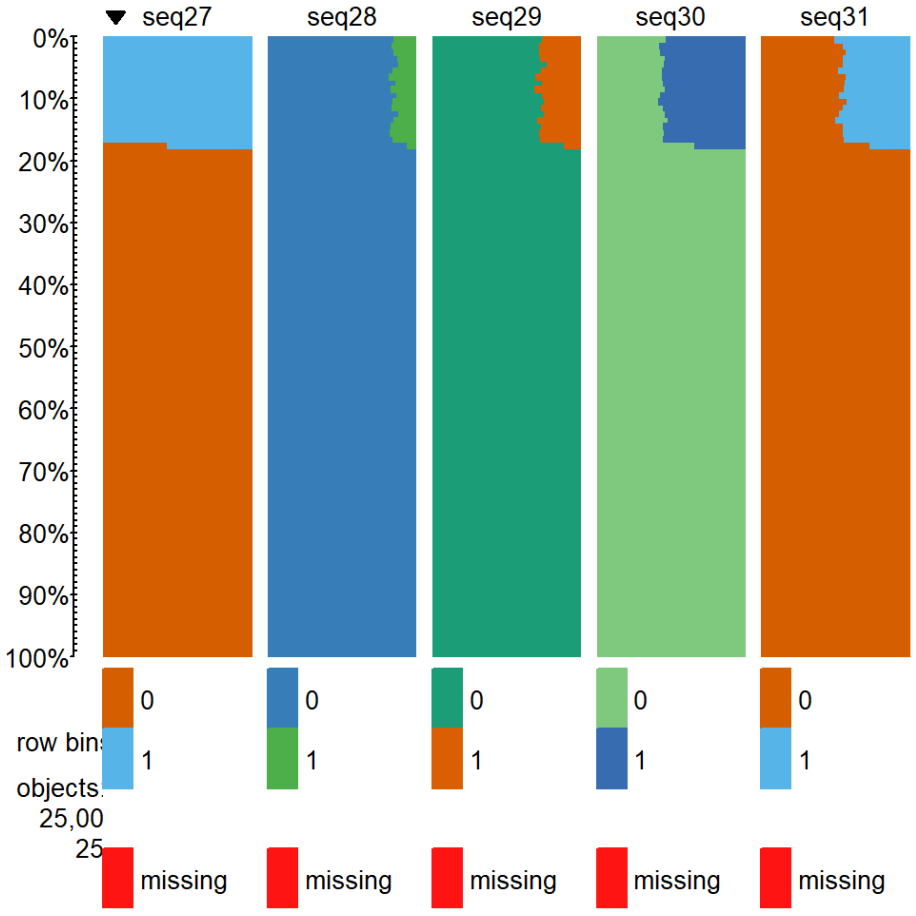
由 seq 系列畫出的相關係數圖可以得知，首先 seq57 無任何變異， 故圖中顯示問號。 seq27-seq31 與可以分成心臟病群組。 seq32、seq33、seq34 三者高度相關。 seq38 、seq39、 seq40，三者高度相關，可以組成癌症群組。 seq42 與 seq43 高度相關，兩者又與 seq49高度相關。 renal、 seq52-54 分為肝炎群組。 seq56與seq58高度相關，可以分成風濕病的群組。

```
pander(prop.table(table(train$dmfail)))
```

0	1
0.365	0.635

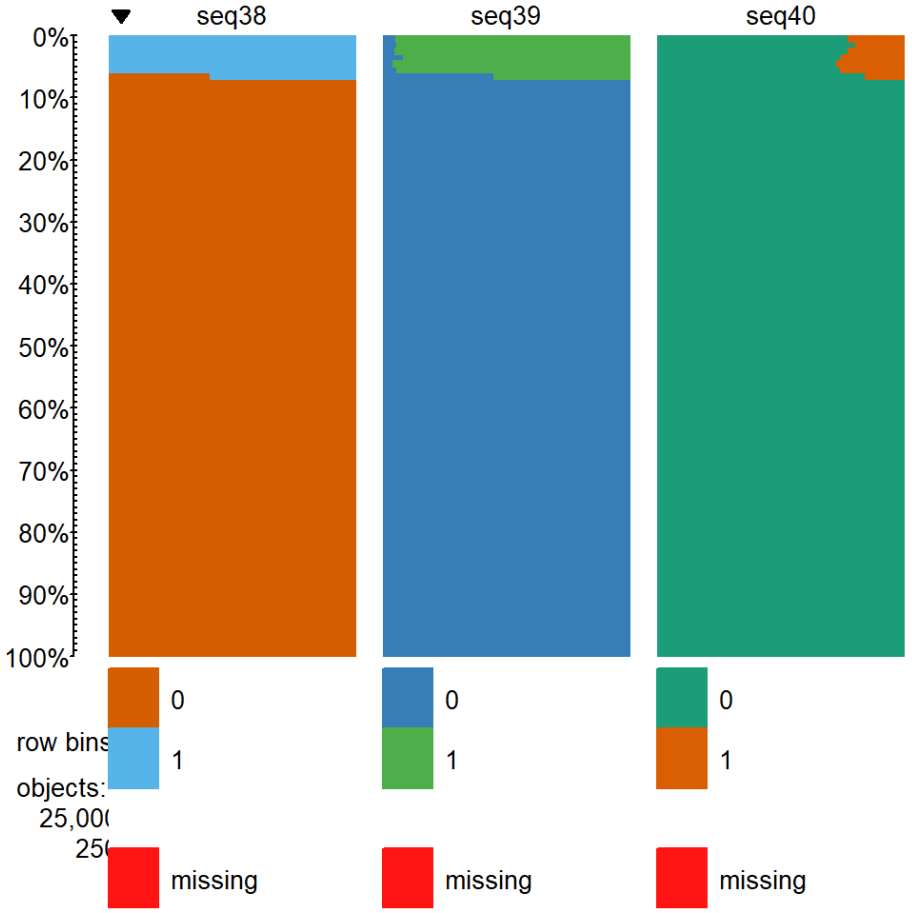
由上表可以知道， 手術成功機率為 0.635， 失敗比率為0.365。 故此資料並非不平衡資料。

```
#seq27        = "Heart Disease" 心臟病
#seq28        = "Myocardial infraction" 心肌梗塞
#seq29        = "Coronary artery disease" 冠狀動脈疾病
#seq30        = "Arrhythmia" 心律失常
#seq31        = "Congestive heart failure" 充血性心力衰竭  ###27=28~31
### seq27 ~seq28 - seq31 #心臟相關
index <- which(names(train)%in% c("seq27","seq28","seq29", "seq30","seq31"))
sapply(train[,index],table)
#>   seq27 seq28 seq29 seq30 seq31
#> 0 20608 24341 23825 22562 22998
#> 1  4392   659  1175  2438  2002
sapply(train[,which(var_names%in%  c("seq28","seq29", "seq30","seq31"))],function(x){table(train$s
eq27,x)})
#>      seq28 seq29 seq30 seq31
#> [1,] 20608 20608 20608 20608
#> [2,]  3733  3217  1954  2390
#> [3,]     0     0     0     0
#> [4,]   659  1175  2438  2002
a<-sapply(train[,which(var_names%in%  c("seq28","seq29", "seq30","seq31"))],function(x){which(x==1
)})
b <- unique(c(a$seq28,a$seq29,a$seq30,a$seq31)) #找出在 seq28~seq31 =1 的值
length(b)==sum(train$seq27==1)
#> [1] TRUE
tableplot(train[,index])
```



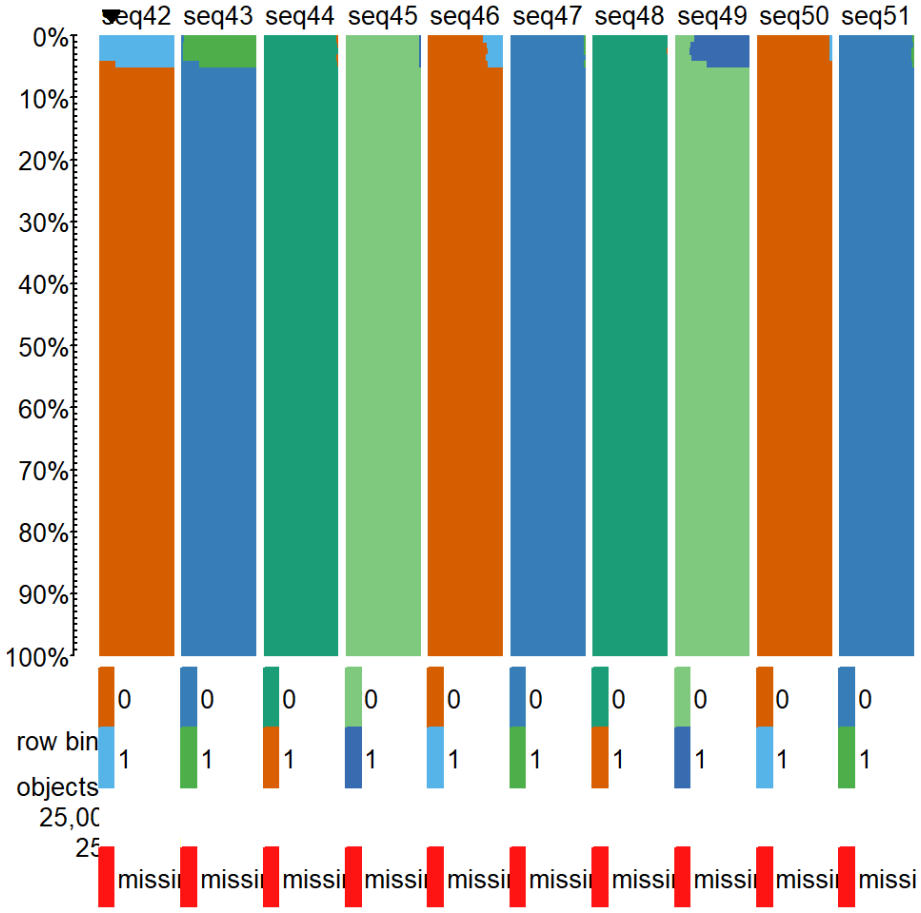
由此表得知， 病患同時得到多種心臟病相關疾病。 心臟病 (seq27)， 是由 心肌梗塞、 冠狀動脈疾病、 心律失常、 充血性心力衰竭 (seq28 -seq31) 等變數組成， 故我們將心臟病變數去除。

```
### seq38 ~seq39 , seq40 #癌症相關
#seq38        = "Cancer"  癌症
#seq39        = "Primary solid cancer" 原發性實體癌
#seq40        = "Metastatistic cancer" 轉移性癌
index <- which(names(train)%in% c( "seq38","seq39", "seq40"))
#pheatmap(as.matrix(train[,index]), cluster_rows = FALSE,  cluster_cols = FALSE)
sapply(train[,which(var_names%in%  c("seq38","seq39", "seq40"))],table)
#>   seq38 seq39 seq40
#> 0 23352 23445 24599
#> 1  1648  1555   401
sapply(train[,which(var_names%in%  c("seq39", "seq40"))],function(x){table(train$seq38,x)})
#>      seq39 seq40
#> [1,] 23352 23352
#> [2,]    93  1247
#> [3,]     0     0
#> [4,]  1555   401
apply(train[,which(var_names%in%  c("seq39", "seq40"))],2,function(x){sum(train$seq38!=x)})
#> seq39 seq40
#>    93  1247
a<-sapply(train[,which(var_names%in%  c("seq39", "seq40"))],function(x){which(x==1)})
b <- unique(c(a$seq39,a$seq40))
sapply(train[,index],table)
#>   seq38 seq39 seq40
#> 0 23352 23445 24599
#> 1  1648  1555   401
length(b)==sum(train$seq38==1)
#> [1] FALSE
tableplot(train[,index])
```



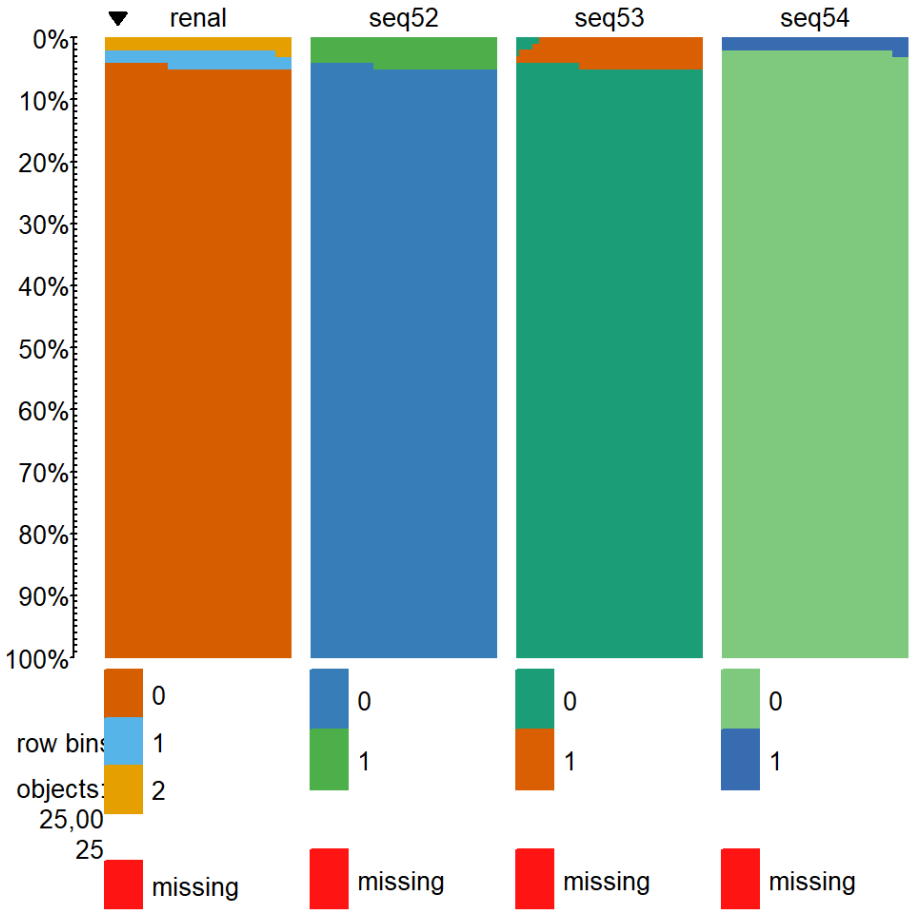
seq38-seq40 是由癌症相關變數組成。 癌症 (seq38)， 有 36 個病患沒有被 原發性實體癌、 轉移性癌 (seq39 - seq40) 包含， 有 63 位病患並沒有得到原發性實體癌、 轉移性癌， 或者是癌症這個變數代表不同意義。

```
#seq42        = "Chronic liver disease"   慢性肝病與肝炎相關
#seq43        = "Heptatitis"  肝炎
#seq44        = "Heptatitis A"
#seq45        = "Heptatitis B"
#seq46        = "Heptatitis C"
#seq47        = "Heptatitis Delta"
#seq48        = "Heptatitis E"
#seq49        = "Heptatitis Others"
#seq50        = "Alcoholic Liver" 酒精性肝病
#seq51        = "Liver Cirrhosis" 肝硬化

###seq42 seq43 -seq49  肝炎
index <- which(names(train)%in% c("seq42","seq43", "seq44", "seq45", "seq46","seq47","seq48", "seq
49","seq50","seq51"))
#pheatmap(train[,index], cluster_rows = FALSE,  cluster_cols = FALSE)
sapply(train[,which(var_names%in%  c("seq42","seq43", "seq44", "seq45", "seq46","seq47","seq48",
"seq49","seq50","seq51"))],table)
#>   seq42 seq43 seq44 seq45 seq46 seq47 seq48 seq49 seq50 seq51
#> 0 23805 23838 24994 24986 24731 24996 24999 24082 24978 24976
#> 1  1195  1162     6    14   269     4     1   918    22    24
sapply(train[,which(var_names%in%  c("seq43", "seq44", "seq45", "seq46","seq47","seq48", "seq49",
"seq50","seq51"))],function(x){table(train$seq42,x)})
#>      seq43 seq44 seq45 seq46 seq47 seq48 seq49 seq50 seq51
#> [1,] 23805 23805 23805 23805 23805 23805 23805 23805 23805
#> [2,]    33  1189  1181   926  1191  1194   277  1173  1171
#> [3,]     0     0     0     0     0     0     0     0     0
#> [4,]  1162     6    14   269     4     1   918    22    24
### seq42 是 seq43~51 的集合體
sapply(train[,which(var_names%in%  c( "seq44", "seq45", "seq46","seq47","seq48", "seq49","seq50`",
"seq51"))],function(x){table(train$seq43,x)})
#>      seq44 seq45 seq46 seq47 seq48 seq49 seq51
#> [1,] 23838 23838 23838 23838 23838 23838 23822
#> [2,]  1156  1148   893  1158  1161   244  1154
#> [3,]     0     0     0     0     0     0    16
#> [4,]     6    14   269     4     1   918     8
### seq43 是 seq44~49 的集合體
sapply(train[,which(var_names%in%  c(  "seq45", "seq46","seq47","seq48", "seq49","seq50","seq51"
))],function(x){table(train$seq44,x)})
#>      seq45 seq46 seq47 seq48 seq49 seq50 seq51
#> [1,] 24980 24731 24990 24993 24077 24972 24971
#> [2,]     6     0     6     6     5     6     5
#> [3,]    14   263     4     1   917    22    23
#> [4,]     0     6     0     0     1     0     1
tableplot(train[,index])
```



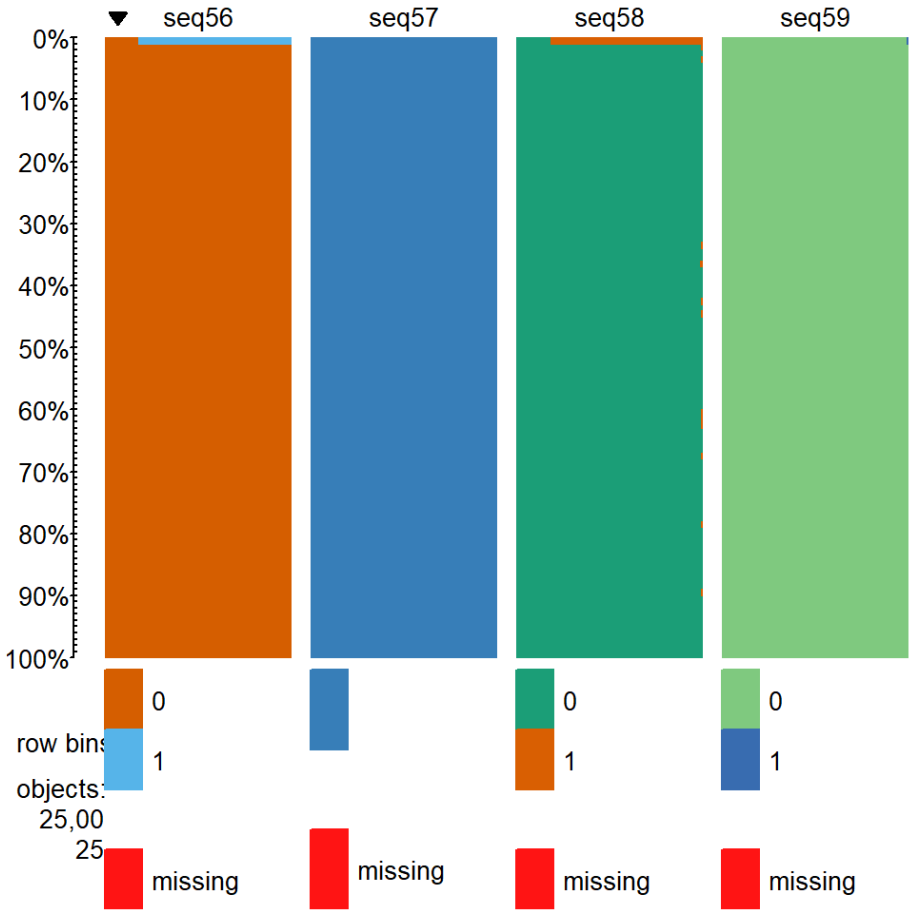
慢性肝病與肝炎相關 (seq42)， 是由 A-E與other肝炎、酒精性肝病、 肝硬化組成 (seq44-seq51)， 而肝炎 (seq43)，是由 A-E 與 other 肝炎組成。 而肝炎中， 只有 C 型 肝炎的患者是較多的， 其他種肝炎得到人數較少， 只有零星幾位。 另外有些病患也有得到複數肝炎的情況， 其中與 C 型肝炎同時得到的只有少數病患。 酒精性肝病、 肝硬化組成的病患相當少， 故不考慮此兩個變數。

```
### renal ~ seq52 seq53 seq54
#seq52        = "Chronic renal disease" 慢性腎病
#seq53        = "End Stage Renal Disease" 晚期腎臟疾病
#seq54        = "Dialysis"  透析

index <- which(names(train)%in% c("renal","seq52","seq53","seq54"))
#pheatmap(train[,index], cluster_rows = FALSE,  cluster_cols = FALSE)
sapply(train[,which(var_names%in%  c("renal","seq52","seq53","seq54"))],table)
#> $renal
#>
#>     0     1     2
#> 23835   645   520
#>
#> $seq52
#>
#>     0     1
#> 23835  1165
#>
#> $seq53
#>
#>     0     1
#> 23891  1109
#>
#> $seq54
#>
#>     0     1
#> 24480   520
sapply(train[,which(var_names%in%  c("seq53","seq54"))],function(x){table(train$seq52,x)})
#>      seq53 seq54
#> [1,] 23835 23835
#> [2,]    56   645
#> [3,]     0     0
#> [4,]  1109   520
apply(train[,which(var_names%in%  c("seq53","seq54"))],2,function(x){sum(train$seq52!=x)})
#> seq53 seq54
#>    56   645
a<-sapply(train[,which(var_names%in%  c("seq39", "seq40"))],function(x){which(x==1)})
b <- unique(c(a$seq39,a$seq40))
sum(train$seq53==train$seq52)
#> [1] 24944
tableplot(train[,index])
```



由上表得知， 腎臟疾病 (renal) 是由慢性腎病 (seq52)與腎臟透析 (seq54)組成， 而特別的是晚期腎臟疾病 (seq53)中， 有 63 人沒有慢性腎臟病。 由於此處變數與腎臟疾病變數資訊重疊， 保留腎臟疾病變數。

```
###0## seq56 ~seq57-59 風濕病相關
#seq56        = "Rheumatoligical Disease" 風濕病
#seq57        = "SLE" 全身性紅斑狼瘡（SLE）
#seq58        = "RA" 類風濕性關節炎
#seq59        = "AS"  僵直性脊椎炎

index <- which(names(train)%in% c("seq56","seq57","seq58","seq59"))
#pheatmap(train[,index], cluster_rows = FALSE,  cluster_cols = FALSE)
sapply(train[,which(var_names%in%  c("seq56","seq57","seq58","seq59"))],table)
table(train$seq56,train$seq58)
tableplot(train[,index])
```



由上表可以得知， 沒有人得到全身性紅斑狼瘡(seq57)， 與僵直性脊椎炎(seq59)， 而少數人得到風濕病(seq56)、 類風溼性關節炎(seq58)， 故不使用這些變數。

```
#seq32        = "Chronic pulmonary disease" 慢性肺病
#seq33        = "COPD" 慢性阻塞性肺病
#seq34        = "Asthma" 哮喘
sum(train$seq33==train$seq32)
table(train$seq34)
```

由上表看出，肺部相關疾病 (seq32-seq34) 慢性肺病 (seq32)與慢性阻塞性肺病(seq33)兩個變數完全相同， 故選擇一個變數使用。 而哮喘則有 1709 人得到。 人數不少故保留。

2 遺失值處理

```
train$city7 <- as.factor(train$city7)

bimp <-train[,-which(names(train) %in% c("seq33","fnf",
                                         "seq56","seq57","seq58","seq59",
                                         "seq43", "seq44", "seq45","seq46",
                                         "seq47", "seq48", "seq49", "seq50","seq51",
                                         "seq52", "seq53","seq54",
                                         "aid19", "aid21" ,
                                         "ch4cat","ch2cat", "ch3cat" ,"cin4cat",
                                         "cin2cat","cin3cat","areacode" ,
                                         "city5cat", "city7cat",
                                         "dmfail","med_cost"))]
Index_imp <-which(names(bimp) %in% c("area4cat","city7","nihno","insamt","paym"))
imp_method <- rep("",dim(bimp)[2])
imp_method[Index_imp] <- c("pmm","polr","pmm","pmm" ,"pmm")
imp <- mice(bimp,imp_method,m=1,maxit=10)
imp2 <-complete(imp)
imp2$dmfail <- train$dmfail
imp2$med_cost <- train$med_cost
```

由上方的資料探索得知， 有許許多多的變數是由其他變數組成的， 故將變數集合或是切割的變數刪除， 並使用 MICE 套件中的 pmm 方法對 area4cat、 nihno、 insamt、 paym進行單一差補。 city7 為順序型變數使用 polr 方法進行差補。

3 醫療花費預測

3.1 模型變數選擇與轉換

```
imp$cindex <- (imp$cindex-0)/(17-0)
imp$cno    <- (imp$cno-0)/(11-0)
imp$insamt <- (imp$insamt-0)/131700
imp$paym   <- (imp$paym-0)/137970
imp$age    <- (imp$age-65)/(106.308-65)
```

在此對連續型變數 cindex, cno, insamt, paym, age, 轉換成 0-1 之間的變數， 以防有變數尺度不一導致參數估計被該變數過度影響。

```
names(imp)[nearZeroVar(imp)]
#> [1] "renal" "seq28" "seq29" "seq36" "seq37" "seq40" "seq43" "seq46" "seq55"
```

由上表得知變異數小的變數， 經由資料探索之後我選擇刪除 seq36、seq37。

```
set.seed(1111)
inTrain <- createDataPartition(y=imp$med_cost, p=0.8, list=FALSE)
train <- imp[inTrain,]
test  <- imp[-inTrain,]
PredVar     <-  setdiff(var_names,
                        c("area4cat","city7","hospvol4cat","ch4cat",
                          "seq30","seq28" ,"seq29", "seq36", "seq37",
                          "seq40","seq42","seq38",  "seq35","seq26",
                          "cno","dmfail","insamt","seq58"))
train.x <- data.frame(train[,c(PredVar)])
train.x <- model.matrix(~.,train.x)
train.y <- imp$med_cost[inTrain]
test.x <- data.frame(test[,c(PredVar)])
test.x <- model.matrix(~.,test.x)
test.y  <- imp$med_cost[-inTrain]
```

```
PredVar
#>  [1] "renal"      "age"        "male"       "bipolar"    "cindex"
#>  [6] "seq25"      "seq27"      "seq31"      "seq32"      "seq34"
#> [11] "seq39"      "seq41"      "seq43"      "seq46"      "seq55"
#> [16] "hospvolume" "nihno"      "paym"       "med_cost"
```

上表為最終的預測變數。

3.2 模型建立方法

```
ctrl <- trainControl(method = "repeatedcv",repeats = 5)
```

此處建立模型的方法， 為了及調整參數避免抽樣誤差， 此處使用五次與十次交叉驗證建模。

```
cv.lm.fit <- train(med_cost ~.,
                   data = train[,PredVar],
                   method = "lm",
                   metric = "RMSE",
                   preProcess=c("center","scale"),
                   trControl =ctrl)

lm.predictions <- predict(cv.lm.fit,test)
LM.RMSE <- RMSE(lm.predictions, test.y)
LM.RMSE
#> [1] 58285.22
```

```
cv.back.lm.fit <- train(med_cost ~., data = train[,PredVar],
                        method = 'leapBackward',
                        metric = "RMSE",
                        trControl =ctrl)

lm.back.predictions <- predict(cv.back.lm.fit,test)
LM.back.RMSE <- RMSE(lm.back.predictions, test.y)
LM.back.RMSE
#> [1] 58550.18
```

```


lambda <- 10^seq(-3,3, length = 1000)
```

而在 LASSO 以及 Ridge 的部分參數設定在 10^{-3} 至 10^{3} 之間， 進行參數比較， 最後選取最好的參數。

```
LASSO.fit <- train(x=train.x,
                   y=train.y,
                   method = "glmnet",
                   trControl = ctrl,
                   metric="RMSE",
                   maximize=FALSE,
                   tuneGrid = expand.grid(alpha = 1, lambda = lambda))

LASSO.predictions <- predict(LASSO.fit,test.x)
LASSO.RMSE <- RMSE(LASSO.predictions, test.y)
LASSO.fit$bestTune
#>     alpha   lambda
#> 832     1 97.94697
LASSO.RMSE
#> [1] 58290.27
```

LASSO 的最佳參數為 λ = 97.94. 當作預測模型。

```
ridge.fit <- train(x=train.x,
                   y=train.y,
                   method = "glmnet",
                   metric="RMSE",
                   maximize=FALSE,
                   trControl = ctrl,
                   tuneGrid = expand.grid(alpha = 0, lambda = lambda)
                   )

ridge.predictions <- predict(ridge.fit,test.x)
ridge.RMSE <- RMSE(ridge.predictions, test.y)
ridge.fit$bestTune
#>      alpha lambda
#> 1000     0   1000
ridge.RMSE
#> [1] 58284.13
```

ridge 在任一 λ 值皆可以的 RMSE 皆相同， 故選擇與 LASSO 相同的 λ

```
pcr.fit <- train(x=train.x,
                 y=train.y,
                 method = 'pcr',
                 metric="RMSE",
                 maximize=FALSE,
                 trControl = ctrl,
                 tuneGrid = expand.grid(ncomp=2:18)
                 )

pcr.predictions <- predict(pcr.fit,test.x)
pcr.RMSE <- RMSE(pcr.predictions, test.y)
pcr.fit$bestTune
#>    ncomp
#> 17    18
pcr.RMSE
#> [1] 58620.49
```

在 pcr 中， 最小的RMSE 在 ncomp = 18， 故選擇 18 個主成分。

```
pls.fit <- train(x=train.x,
                 y=train.y,
                 method = 'pls',
                 metric="RMSE",
                 maximize=FALSE,
                 tuneGrid = expand.grid(ncomp=2:18),
                 trControl = ctrl
)

pls.predictions <- predict(pls.fit,test.x)
pls.RMSE <- RMSE(pls.predictions, test.y)
pls.fit$bestTune
#>    ncomp
#> 17    18
pls.RMSE
#> [1] 58285.11
```

在 plsr 中， 最小的RMSE 在 ncomp = 18， 故選擇 18 個主成分。

```
kernelpls.fit <- train(x=train.x,
                       y=train.y,
                       method = 'kernelpls',
                       metric="RMSE",
                      maximize=FALSE,
                      trControl = ctrl,
                      tuneGrid = expand.grid(ncomp=2:18)
                      )

kernelpls.predictions <- predict(kernelpls.fit,test.x)
kernelpls.RMSE <- RMSE(kernelpls.predictions, test.y)
kernelpls.fit$bestTune
#>    ncomp
#> 16    17
kernelpls.RMSE
#> [1] 58282.48
```

在 kenerelpls 中， 最小的RMSE 在 ncomp = 17， 故選擇 17 個主成分。

```
rpart.fit <- train(x=train.x,
                   y=train.y,
                   method = 'rpart',
                   metric="RMSE",
                   maximize=FALSE,
                   trControl = ctrl
                   )

rpart.predictions <- predict(rpart.fit,test.x)
rpart.RMSE <- RMSE(rpart.predictions, test.y)
rpart.fit$bestTune
#>           cp
#> 1 0.01125474
rpart.RMSE
#> [1] 58968.56
```

在 CART 的決策樹， 選擇 cp= 0.01125474 ， 作為最佳剪枝參數。

```
rpart1SE.fit <- train(x=train.x,
                      y=train.y,
                      method = 'rpart1SE',
                      metric="RMSE",
                      maximize=FALSE,
                      trControl = ctrl
                      )

rpart1SE.predictions <- predict(rpart1SE.fit,test.x)
rpart1SE.RMSE <- RMSE(rpart1SE.predictions, test.y)
rpart1SE.RMSE
#> [1] 58640.47
```

在 CART (rpart1SE) 的決策樹， 選擇最大深度為 3， 作為最佳剪枝參數。

```
rpart2.fit <- train(
  x=train.x,                    y=train.y, method = 'rpart2',
  metric="RMSE",
  maximize=FALSE,
  trControl = ctrl,
  tuneGrid = expand.grid(maxdepth=1:7)
)

rpart2.predictions <- predict(rpart2.fit,test.x)
rpart2.RMSE <- RMSE(rpart2.predictions, test.y)
rpart2.fit$bestTune
#>   maxdepth
#> 3        3
rpart2.RMSE
#> [1] 58640.47
```

在 CART (rpart2) 的決策樹， 選擇最大深度為 3， 作為最佳剪枝參數。

```
Gam.fit <- train(train.x,
                 train.y,
                 method = 'gam',
                 metric="RMSE",
                 maximize=FALSE,
                 trControl = ctrl)

Gam.predictions <- predict(Gam.fit,test.x)
Gam.RMSE <- RMSE(Gam.predictions, test.y)
Gam.RMSE
#> [1] 58366.28
```

LM	LM.back	LASSO	Ridge	Gam	PCR	PLS	Kernelpls	rpart	rpart1SE	rpart2
58285	58550	58290	58284	58366	58620	58285	58282	58969	58640	58640

由上表得知， 各個方法表現都差不多， 線性回歸、 LASSO、 Ridge、 PLS、 Kernelpls、 GAM 的 RMSE 差不多。 經由與組員討論之後， 最後選擇 GAM 當作我們最終預測模型。

4 類別預測模型建立方法

此處建立模型的方法， 為了及調整參數避免抽樣誤差， 此處使用五次與十次交叉驗證建模。

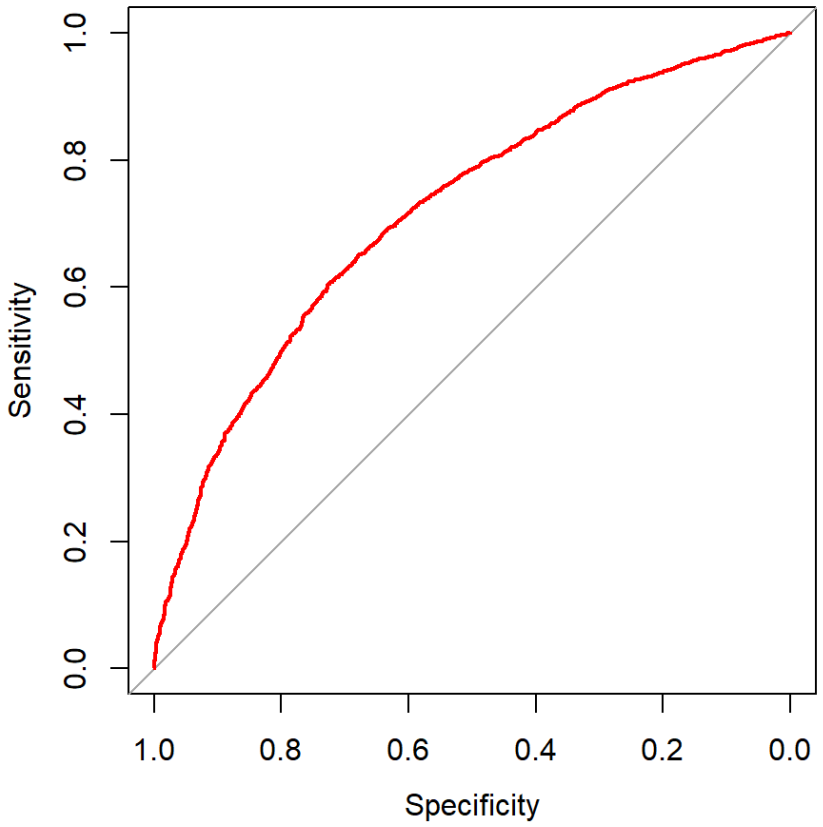
4.1 Logit

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.684	0.42	0.836	0.596	0.715	0.724	0.446

```
lambda <- 10^seq(-3,3, length = 1000)
```

而在 LASSO 以及 Ridge 的部分參數設定在 10^{-3} 至 10^{3} 之間， 進行參數比較， 最後選取最好的參數。

4.2 Ridge

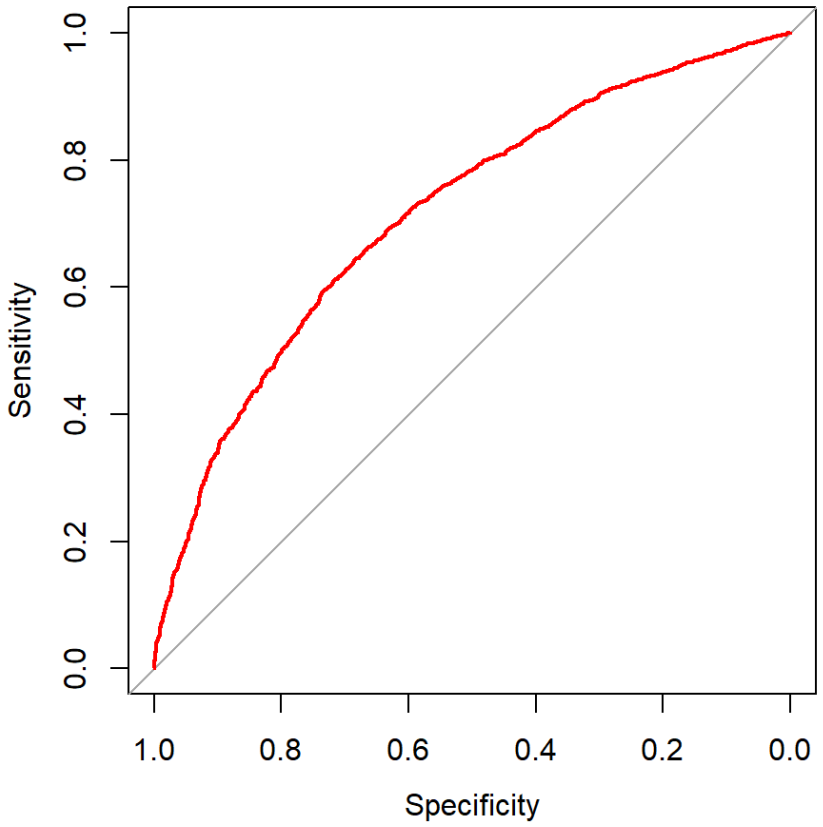



```
#>   alpha lambda
#> 1     0      0
```

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.681	0.378	0.856	0.601	0.705	0.717	0.449

由上表可以得知， Ridge 在 $\lambda = 0$ 時， Accuracy 最高。

4.3 LASSO




```
#>   alpha lambda
#> 1     1      0
```

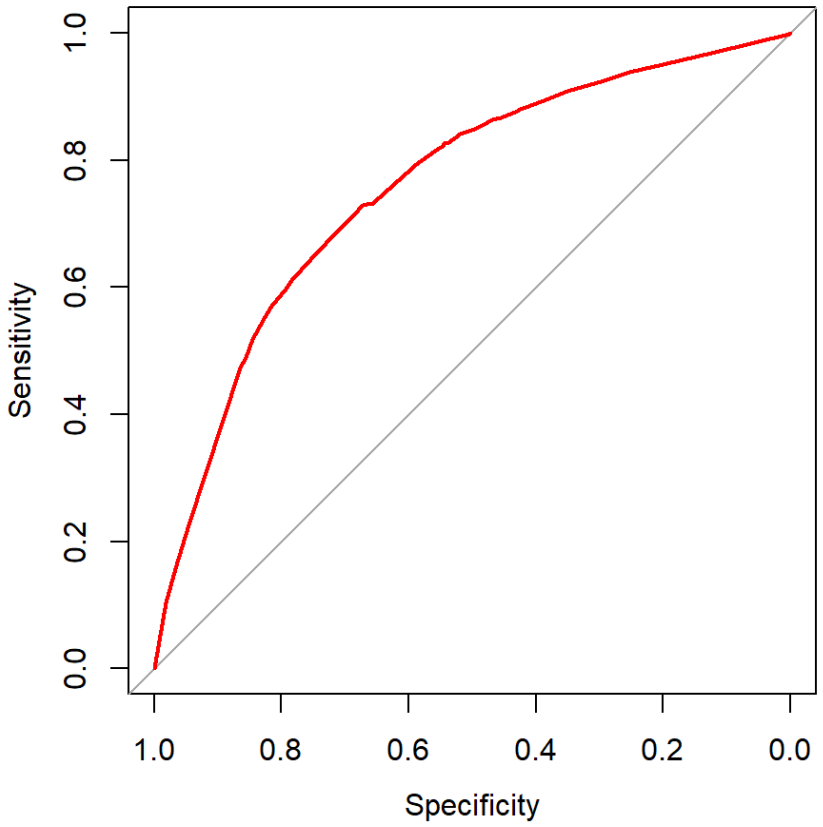
Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.681	0.385	0.851	0.598	0.706	0.716	0.449

由上表可以得知， LASSO 在 $\lambda = 0$ 時， Accuracy 最高。

4.4 LDA

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.687	0.413	0.844	0.604	0.714	0.722	0.447

4.5 CART




```
#>            cp
#> 4 0.001049797
```

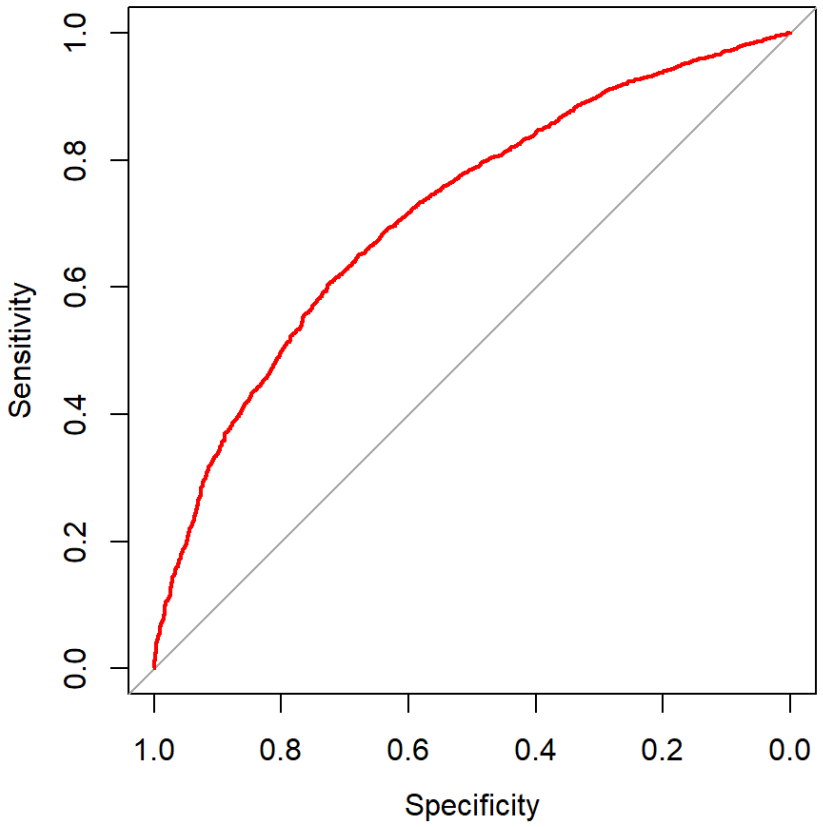
Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.721	0.498	0.849	0.655	0.746	0.757	0.433

由上表得知， CART 在 cp= 0.001 時， 有高的 Accuracy。

4.6 Random Forest

	Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
rf.result	0.751	0.534	0.805	0.611	0.75	0.752	0.439

4.7 KNN

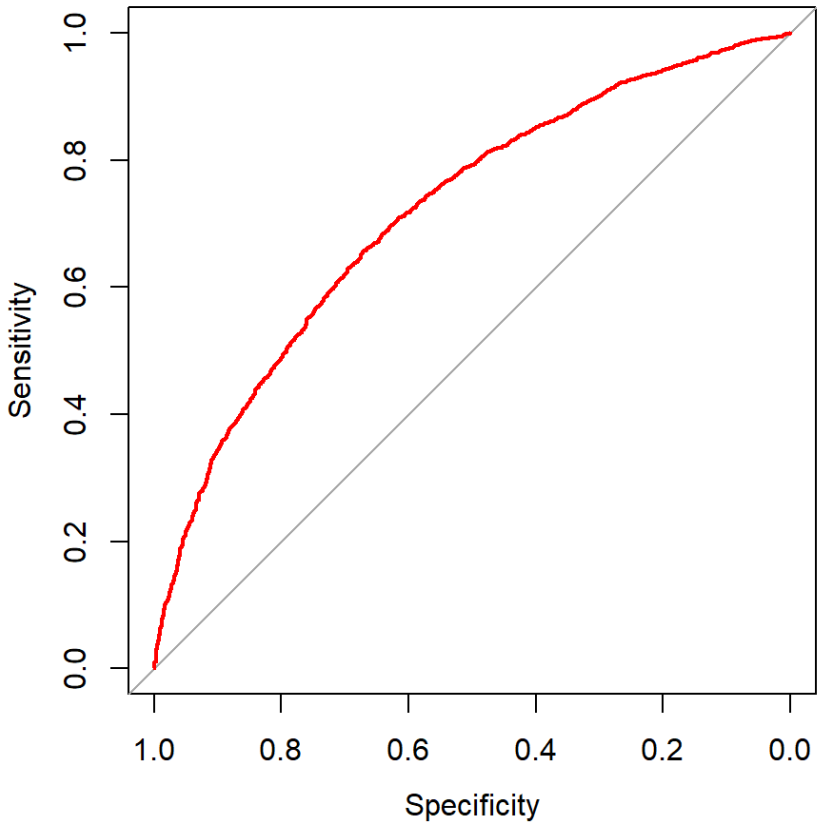



```
#>     k
#> 20 43
```

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.628	0.185	0.882	0.475	0.653	0.717	0.449

由上表得知， knn 在 k = 43 時， 有最高的 Accurach。

4.8 Neural Network




```
#>    size decay
#> 15    3   0.1
```

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.687	0.421	0.84	0.602	0.716	0.718	0.448

由上表得知， 類神經網路在 3 個神經源且衰退在 0.1時， 有最高的 Accuracy。

4.9 Bagging

```
#>   parameter
#> 1      none
```

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.695	0.521	0.795	0.593	0.742	0.74	0.447

4.10 Boosting

```
#>   n.trees interaction.depth shrinkage n.minobsinnode
#> 8     100                 3       0.1             10
```

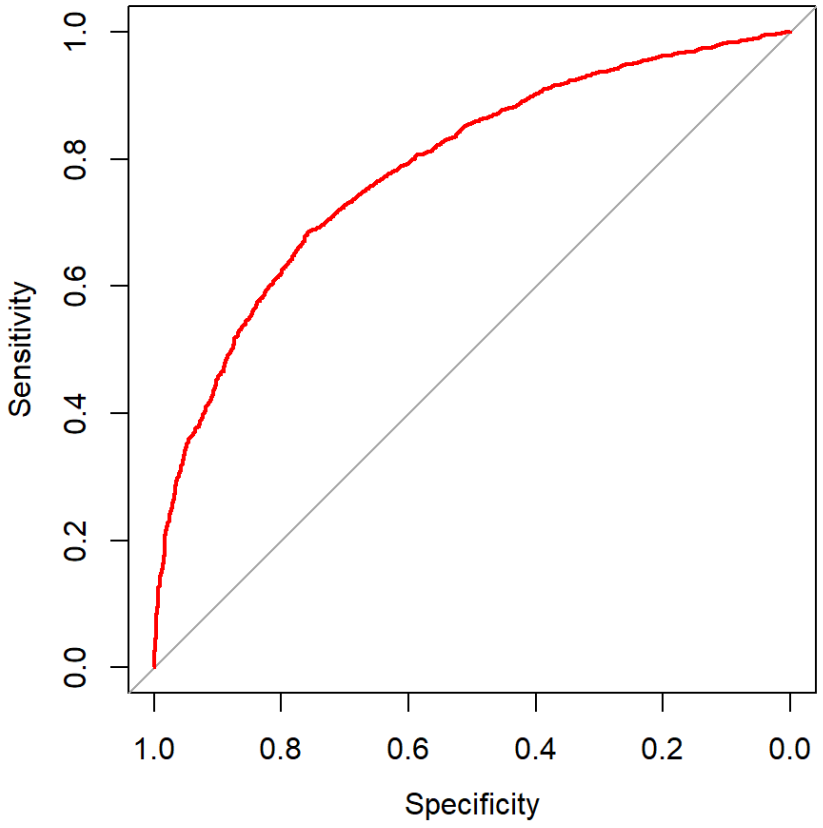
Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.724	0.516	0.844	0.655	0.752	0.781	0.425

由上表得知， boosting 是所有分類技術中表現最好的。 因此我們將選擇此模型做最後模型。

	Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
Logit	0.684	0.420	0.836	0.596	0.715	0.724	0.446
Ridge	0.681	0.378	0.856	0.601	0.705	0.717	0.449
LASSO	0.681	0.385	0.851	0.598	0.706	0.716	0.449
LDA	0.687	0.413	0.844	0.604	0.714	0.722	0.447
Random Forest	0.751	0.534	0.805	0.611	0.750	0.752	0.439
CART	0.721	0.498	0.849	0.655	0.746	0.757	0.433
KNN	0.628	0.185	0.882	0.475	0.653	0.717	0.449
Neural Network	0.687	0.421	0.840	0.602	0.716	0.718	0.448
Bagging	0.695	0.521	0.795	0.593	0.742	0.740	0.447
Boosting	0.724	0.516	0.844	0.655	0.752	0.781	0.425

上表為統整表， 我們可以發現 Boosting 與 CART 的準確率較高， AUC 兩者相較於其他的分類技術是較好的。 綜合準確率與 AUC 的評估下， 我們選擇 Boosting， 作為最終預測模型。

5 不平衡資料處理

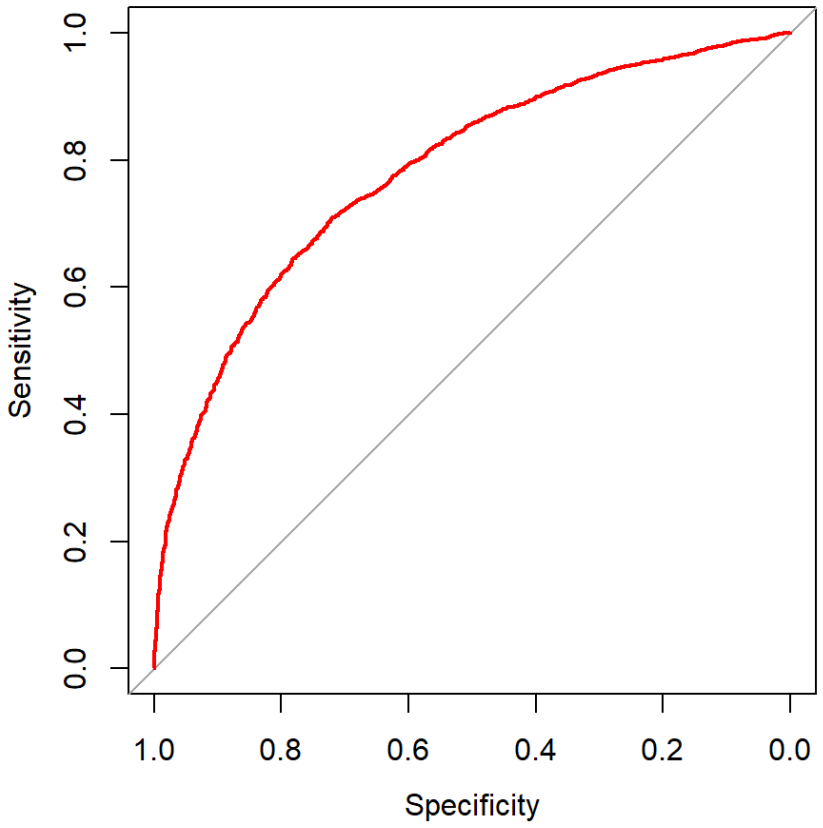



```
#> [1] 0.5254608
#>   n.trees interaction.depth shrinkage n.minobsinnode
#> 9     150                 3       0.1             10
```

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.712	0.759	0.685	0.581	0.832	0.784	0.435

由上表可以知道， Oversampling 的最佳切點為 0.525， 其表現與原資料相差不遠。




```
#> [1] 0.4952382
#>   n.trees interaction.depth shrinkage n.minobsinnode
#> 6     150                 2       0.1             10
```

Accuracy	Sensitivity	Specificity	Pos Pred Value	Neg Pred Value	AUC	RMSE
0.713	0.721	0.709	0.588	0.815	0.781	0.437

由上表可以知道， downampling 的最佳切點為 0.495， 其表現與原資料亦相差不遠。

6 附錄

遭遇問題:

1. 不知為何在 LASSO 與 LDA 的 ROC 圖放在 chunck 中會變成超連結。
2. 做完不平衡資料之後， 如何將切點轉換回原始資料? 直接使用該抽樣的方法的最佳切點套回原始資料就可以了嗎?