R Assignment 4

12111603 谭致恒

2023-12-08

Question 1

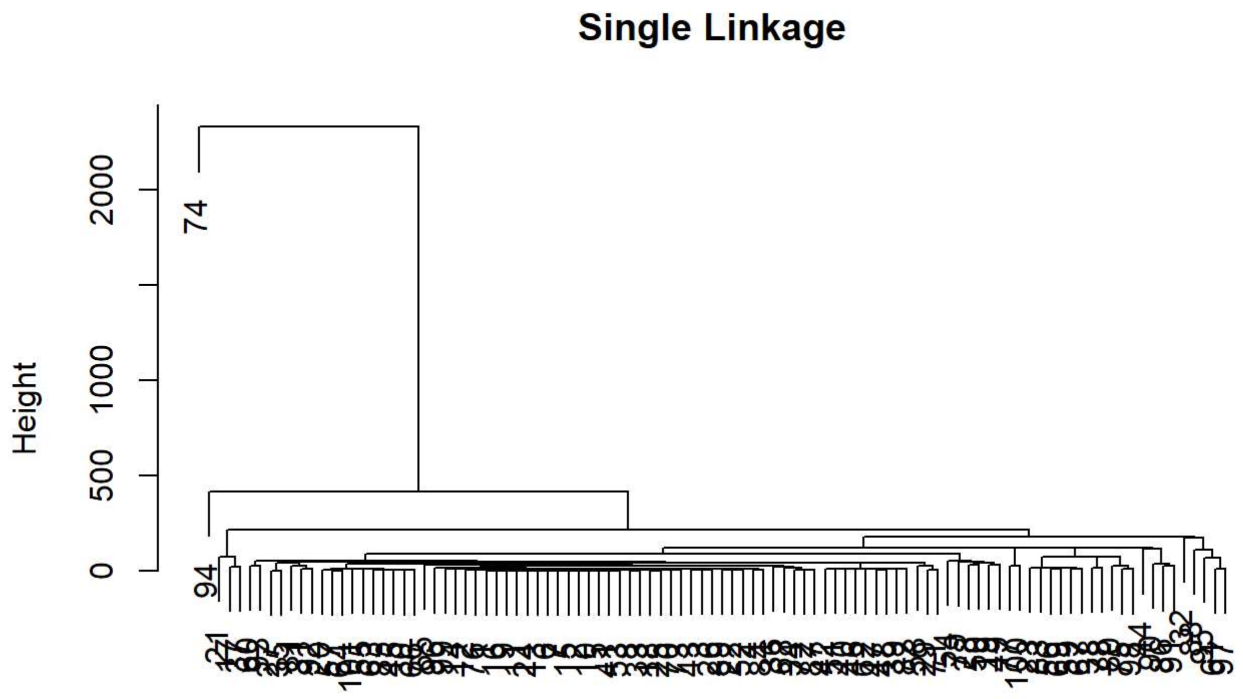
(a)

```
library(lattice)
library(HSAUR3)
```

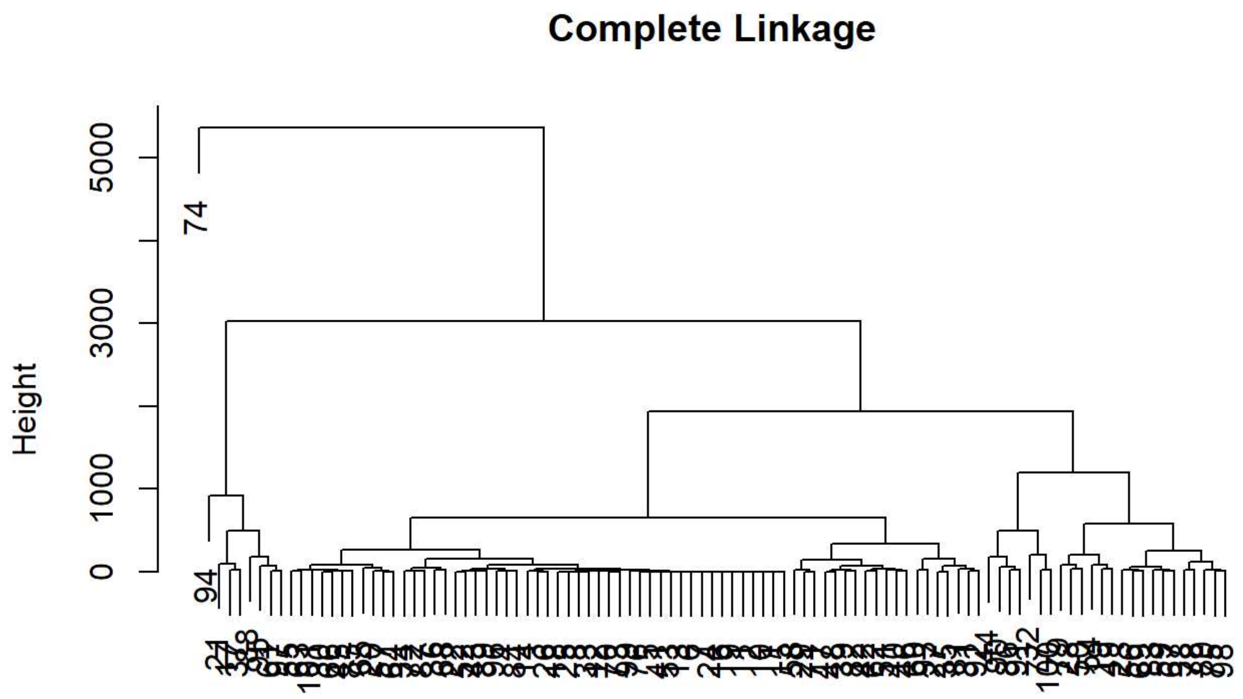
```
## Warning: 程辑包'HSAUR3'是用R版本4.3.2 来建造的
```

```
## 载入需要的程辑包：tools
```

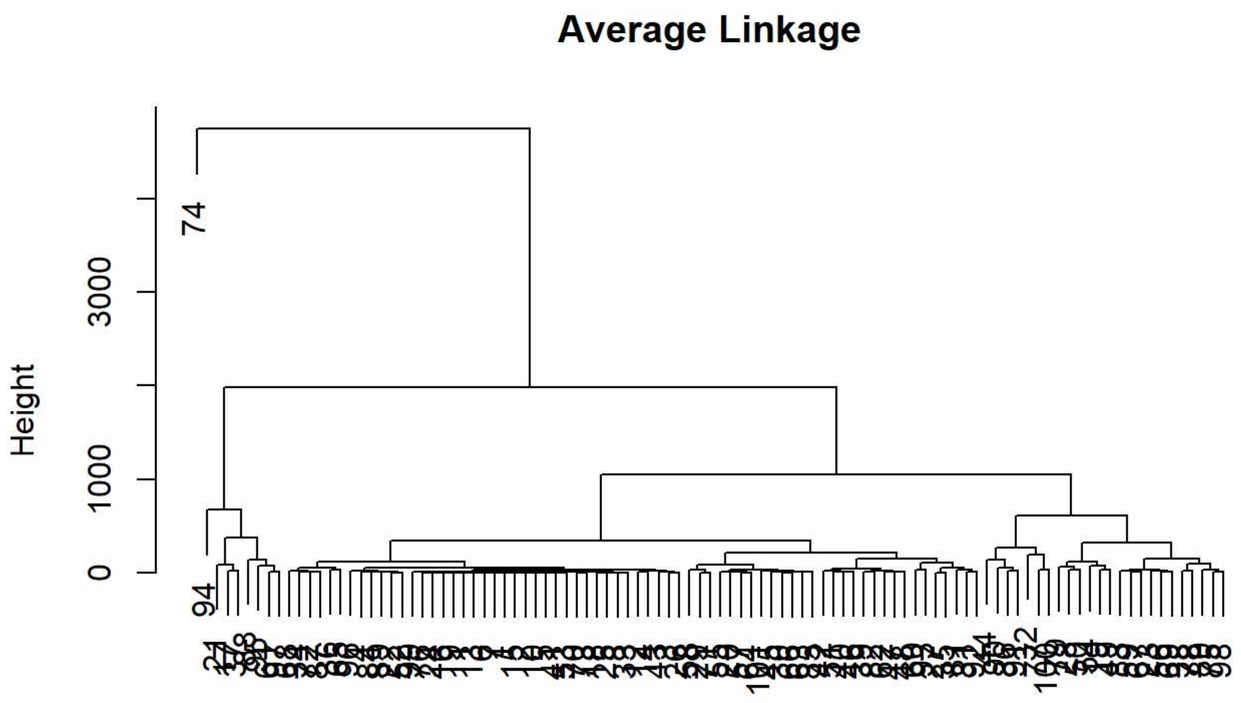
```
data("planets")
distance <- dist(planets)

single <- hclust(distance, method = "single")
complete <- hclust(distance, method = "complete")
average <- hclust(distance, method = "average")
par(mfrow = c(1,1))
plot(single,main="Single Linkage",sub="",xlab="")
```



```
plot(complete,main="Complete Linkage",sub="",xlab="")
```

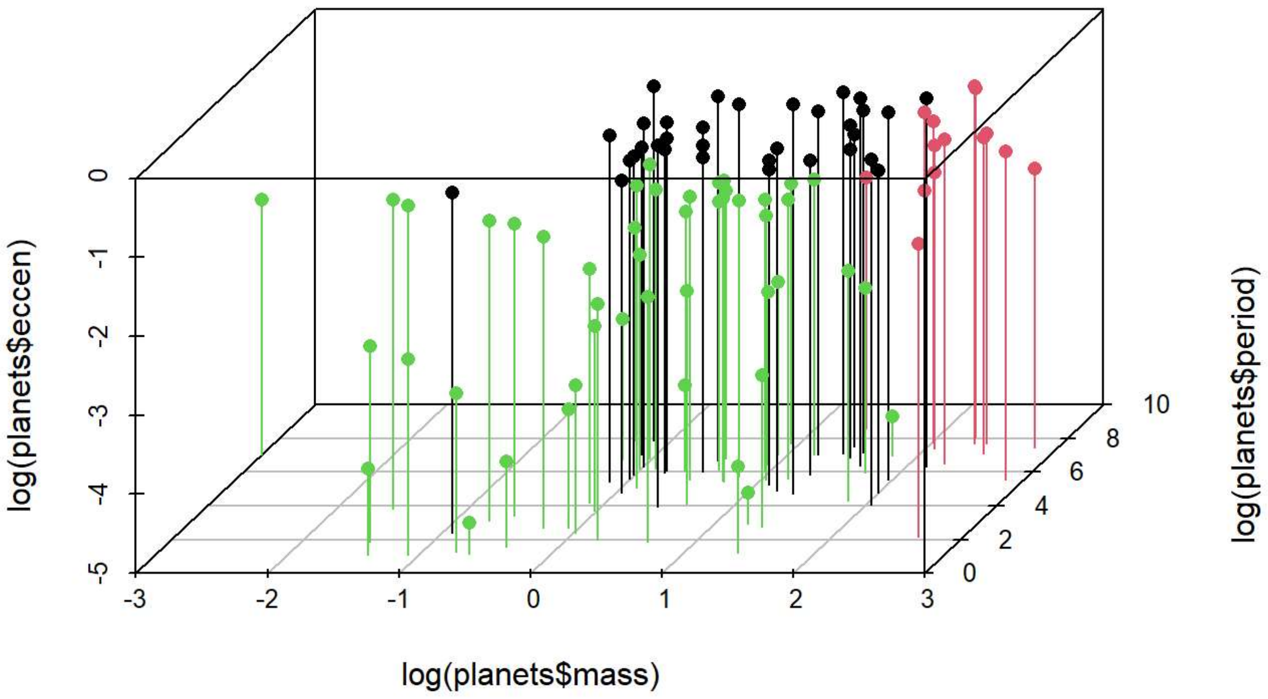


```
plot(average,main="Average Linkage",sub="",xlab="")
```

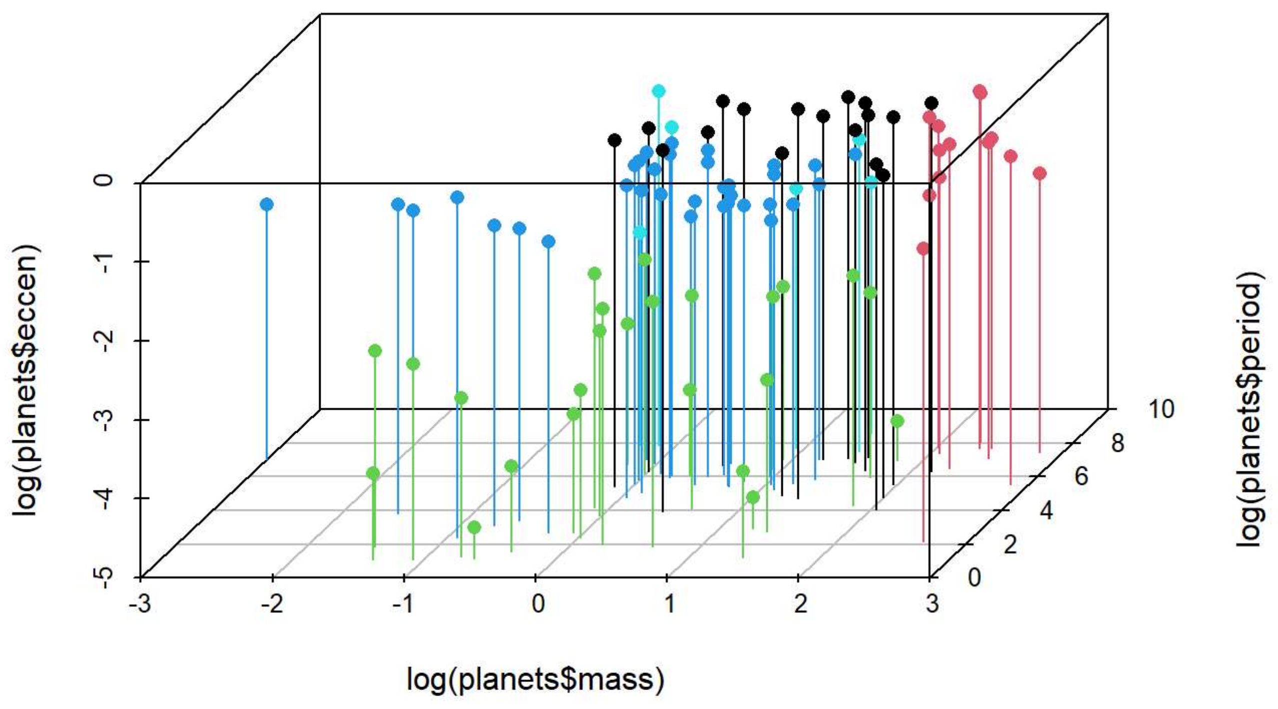


(b)

```
library("scatterplot3d")
scaled <- scale(planets)
set.seed(123)
kmeans3= kmeans(scaled,centers=3)
scatterplot3d(log(planets$mass), log(planets$period), log(planets$eccen),
              color = kmeans3$cluster, type = "h", angle = 55,
              scale.y = 0.7, pch = 16, y.ticklabs = seq(0,10, by = 2),
              y.margin.add = 0.1)
```



```
set.seed(123)
kmeans5= kmeans(scaled,centers=5)
scatterplot3d(log(planets$mass), log(planets$period), log(planets$eccen),
              color = kmeans5$cluster, type = "h", angle = 55,
              scale.y = 0.7, pch = 16, y.ticklabs = seq(0,10, by = 2),
              y.margin.add = 0.1)
```



(c)

```
summary(planets$eccen)
```

```
##    Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
##  0.0000  0.1000  0.2700  0.2815  0.4100  0.9270
```

We discover that there are zeros in the data, so we first Shift up one unit for the whole data to solve the problem caused by zero data. Correspondingly in this case, the fitted $\hat{\mu}_1$ and $\hat{\mu}_2$ is bigger than their expected estimators than 1 while other estimators are not affected.

```
logL <- function(param, x) {
   d1 <- dnorm(x, mean = param[2], sd = param[3])
   d2 <- dnorm(x, mean = param[4], sd = param[5])
   -sum(log(param[1] * d1 + (1 - param[1]) * d2))
}

startparam <- c(p = 0.5, mu1 = 0.3, sd1 = 1, mu2 = 0.5,sd2 = 1)
set.seed(123)
opp <- optim(startparam, logL, x = planets$eccen + 1,
                method = "L-BFGS-B", lower = c(0.01, rep(0.001, 4)), upper = c(0.99, rep(2,
4)))
opp
```

```
## $par
##          p        mu1        sd1        mu2        sd2
## 0.81018255 1.34177945 0.18742088 1.02446519 0.02447579
##
## $value
## [1] -27.81359
##
## $counts
## function gradient
##       95       95
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Thus, the model comes to

$$f(x) = \phi\mathcal{N}^{(1)}(x, \mu_1, \sigma_1^2) + (1-\phi)\mathcal{N}^{(2)}(x, \mu_2, \sigma_2^2)$$

where $\phi = 0.8101825$, $\mu_1 = 0.3417795$, $\sigma_1^2 = 0.0351266$, $\mu_2 = 0.0244652$ and $\sigma_2 =$ 5.9906418^{-4}.

(d)

```
library(mclust)
```

```
## Warning: 程辑包'mclust'是用R版本4.3.2 来建造的
```

```
## Package 'mclust' version 6.0.0
## Type 'citation("mclust")' for citing this R package in publications.
```

```
set.seed(123)
result <- Mclust(planets$eccen)
result$parameters
```

```
## $pro
## [1] 0.1909587 0.8090413
##
## $mean
##          1          2
## 0.02474142 0.34216167
##
## $variance
## $variance$modelName
## [1] "V"
##
## $variance$d
## [1] 1
##
## $variance$G
## [1] 2
##
## $variance$sigmasq
## [1] 0.000609509 0.035071269
##
## $variance$scale
## [1] 0.000609509 0.035071269
```

Thus, the model comes to

$$f(x) = \phi\mathcal{N}^{(1)}(x, \mu_1, \sigma_1^2) + (1-\phi)\mathcal{N}^{(2)}(x, \mu_2, \sigma_2^2)$$

where $\phi = 0.1909587$, $\mu_1 = 0.0247414$, $\sigma_1^2 =$ 6.0950903^{-4}, $\mu_2 = 0.3421617$ and $\sigma_2 = 0.0350713$.

The result in part (d) is consistent with that in part (c).

(e)

```
set.seed(123)
pca <- prcomp(planets, scale = TRUE)
```

The coefficients for the first two principal components are listed as follows.

```
pca$rotation[ ,1:2]
```

```
##              PC1         PC2
## mass   0.6423065 -0.05996314
## period 0.5229950  0.76306298
## eccen  0.5602843 -0.64353657
```

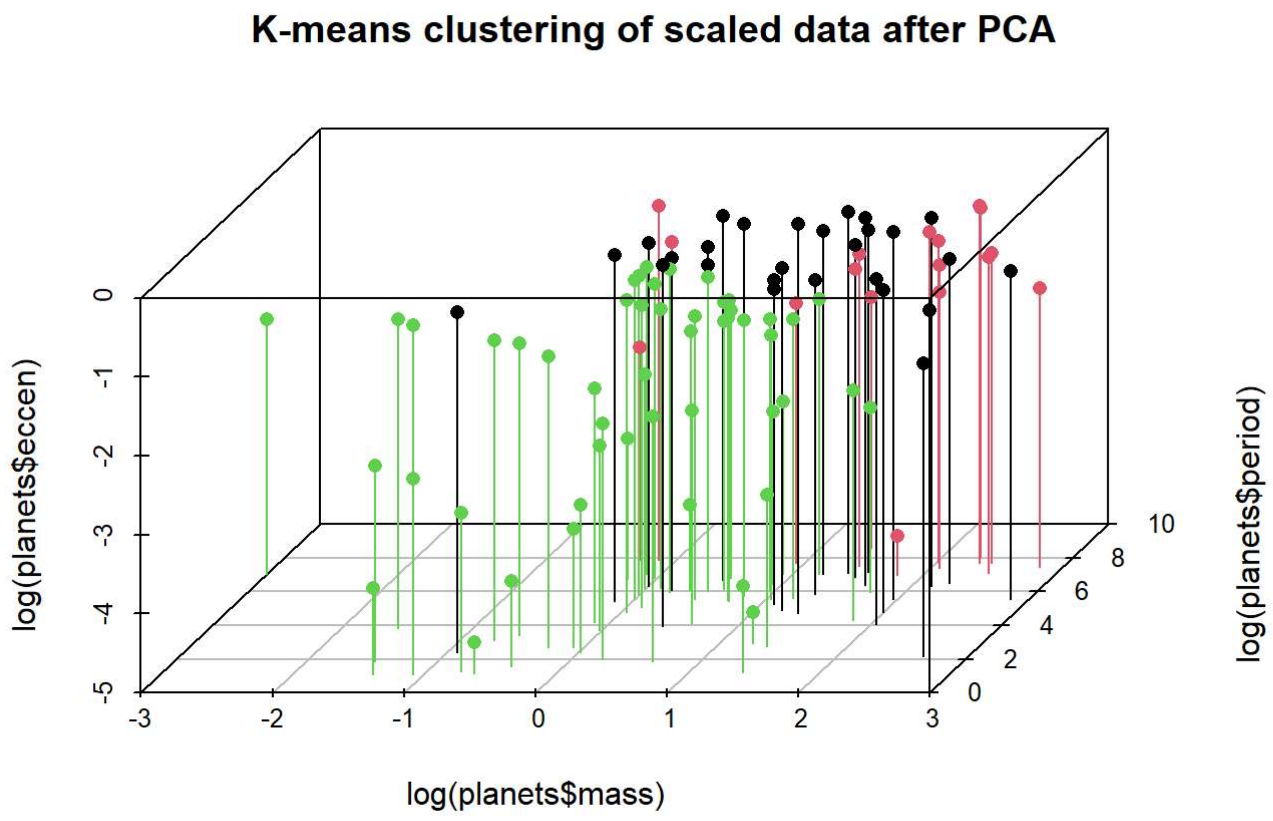
The principal component scores for each planet are listed as follows.

```
pca$x
```

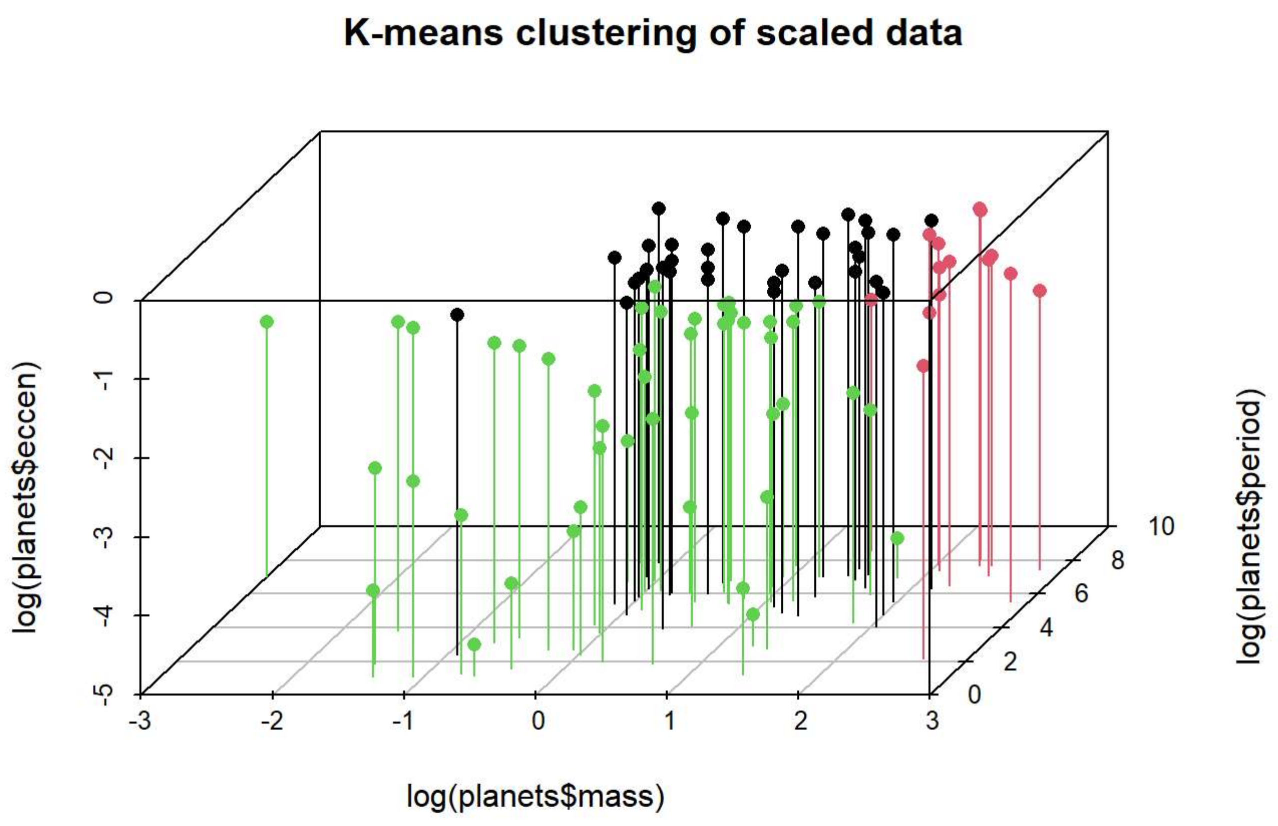
```
##             PC1          PC2          PC3
## 1  -1.703521406  0.333412263 -0.317723645
## 2  -1.690670027  0.331302824 -0.334134464
## 3  -0.761204110 -0.670962589  0.521211858
## 4  -0.899957517 -0.460553244  0.384507035
## 5  -1.470968444  0.088828815 -0.142157505
## 6  -1.628866016  0.268597257 -0.296106234
## 7  -1.453818089  0.084051724 -0.166479222
## 8  -0.328364590 -1.185227634  0.857857639
## 9  -1.652030119  0.327266919 -0.380630115
## 10 -1.642874006  0.327080514 -0.390697571
## 11 -1.508768650  0.173731908 -0.269489321
## 12 -0.833611684 -0.572710156  0.356629745
## 13 -1.604774912  0.294443962 -0.381000325
## 14 -0.894526937 -0.475478120  0.269347580
## 15 -1.552872634  0.262983037 -0.384682573
## 16 -1.605775861  0.322962403 -0.435638183
## 17  0.223486270  2.278978687  0.921884194
## 18 -0.867342852 -0.492706556  0.218879861
## 19 -0.983648497  0.919756324 -0.003954419
## 20 -0.006926677 -1.137171697  0.953613065
## 21  1.526980544  0.685274524  2.100008258
## 22 -1.188140002 -0.023619320 -0.130265394
## 23 -0.013888070  0.301724319  0.827731460
## 24 -1.563279363  0.318593381 -0.486688478
## 25 -1.476672551  0.237173332 -0.407496711
## 26 -1.344904490  0.136129674 -0.310136647
## 27 -0.989331825  0.447137596 -0.035646395
## 28 -0.288827708  0.222501340  0.570830607
## 29 -0.341819216 -0.550128939  0.583362494
## 30 -0.385041510 -0.613191358  0.551978404
## 31  0.153460192 -0.799860208  1.015739803
## 32  0.673676747  0.790275183  1.328250468
## 33 -1.426139647  0.189555649 -0.393283377
## 34 -0.542969893 -0.649376687  0.404283622
## 35 -0.621790925  0.047784171  0.258425689
## 36 -0.614183042 -0.497909017  0.304088800
## 37  0.037908210  2.595166702  0.602398503
## 38 -0.347228837  1.041922025  0.372095526
## 39 -0.214558142 -0.567990814  0.606402616
## 40 -0.055057819 -0.681578440  0.749393536
## 41 -1.143520921 -0.092419140 -0.226744783
## 42 -1.171509632  0.618929291 -0.355165237
## 43  0.297759588 -1.629074895  1.010101238
## 44 -0.059345034  0.956988939  0.503417867
## 45 -0.658610158 -0.199814272  0.083340902
## 46 -0.087854829 -0.558042165  0.573455105
## 47  0.209516225 -0.958005671  0.851258877
## 48 -0.243206505 -0.443819721  0.426790238
## 49  0.848768694 -1.075064375  1.383582089
## 50 -0.348565080  0.480827967  0.226549467
## 51 -0.605104445  0.087566988  0.016871857
## 52 -1.095466131  0.048467670 -0.413595699
## 53 -0.993684420 -0.152041949 -0.315857932
## 54  0.711359076 -0.943966840  1.147558254
## 55  -0.587782978 -0.223802437 -0.022359296
## 56  -0.597372288  0.788846400  0.601262625
## 57  -0.420207682 -0.430700452  0.124074829
## 58  -0.565848109 -0.104750398 -0.080822730
## 59  -0.472965629  1.057663456 -0.200638924
## 60  -0.749298605  0.654486666 -0.401286743
## 61   0.503535941  1.648568524  0.570287524
## 62  -0.462877062  0.009093324 -0.141872980
## 63  -0.577216620 -0.076100630 -0.305972798
## 64  -0.147121897 -0.583840779  0.059855439
## 65   0.002211130 -0.805905002  0.170810635
## 66  -0.377211666 -0.191366341 -0.299477970
## 67   0.100494738  0.564043567  0.028580979
## 68   0.297972212 -1.166022173  0.326794165
## 69   0.919661797 -0.340259408  0.759698104
## 70  -1.070455711  0.258595779 -0.999317069
## 71   0.097165709 -0.532369568 -0.048184675
## 72   1.482287080 -2.462896010  1.235655294
## 73  -0.897754987  0.139323959 -1.004792716
## 74   2.603895761  4.458750841  1.599748890
## 75   1.442220277 -0.662437726  0.982456118
## 76  -0.949843973  0.205475045 -1.103550893
## 77   1.011203195  0.725098846  0.457916064
## 78   0.627835835  0.458158282  0.149491677
## 79   0.822348900 -0.284580296  0.296616717
## 80   1.218537242 -0.237997345  0.519327049
## 81   1.407137925 -1.416349503  0.635219323
## 82  -0.317535024  0.359110502 -1.138782140
## 83   0.005677719  1.136378075 -1.063202584
## 84  -0.256459179 -0.088240546 -1.262835911
## 85   1.536314165 -1.697724708  0.058291570
## 86   1.778750857  0.133980423  0.040114637
## 87   0.699735796 -0.908224948 -0.795792285
## 88   2.007260665  1.012444470  0.124150192
## 89   1.056167995 -1.357496410 -0.560477228
## 90   1.213952042  0.945137698 -0.697437474
## 91   1.435238879  0.603406441 -0.502383447
## 92   1.995773521 -1.494857341 -0.094099930
## 93   1.445323958 -0.507323990 -1.011218948
## 94   3.318745016  1.105864335  0.330322879
## 95   2.995113271  0.117878297  0.004245738
## 96   1.112024683 -0.780876060 -1.717950563
## 97   2.457878305  1.021389082 -0.848946457
## 98   2.069636084  0.062934791 -1.341881538
## 99   1.526318501 -0.741254008 -2.595981508
## 100  2.868605488  0.879282917 -2.483757420
## 101  2.619234399 -1.039237215 -2.756168649
```

(f)

```
scaled2 <- scale(pca$x[ ,1:2])
par(mfrow =c(1,1))
set.seed(123)
pca_kmeans3= kmeans(scaled2,centers=3)
scatterplot3d(log(planets$mass), log(planets$period), log(planets$eccen),
              color = pca_kmeans3$cluster, type = "h", angle = 55,
              scale.y = 0.7, pch = 16, y.ticklabs = seq(0,10, by = 2),
              y.margin.add = 0.1, main = "K-means clustering of scaled data after PCA")
```



```
scatterplot3d(log(planets$mass), log(planets$period), log(planets$eccen),
              color = kmeans3$cluster, type = "h", angle = 55,
              scale.y = 0.7, pch = 16, y.ticklabs = seq(0,10, by = 2),
              y.margin.add = 0.1, main = "K-means clustering of scaled data")
```



The results of the two are generally the same, with only a small amount of inconsistency.

Question 2

(a)

```
library(ISLR)
data(Default)

set.seed(12345)
train <- sample(nrow(Default), 0.7*nrow(Default))
training <- Default[train, ]
validation <- Default[-train, ]

log.model <- glm(default ~ student + balance + income, data = training, family = binomial())

prob <- predict(log.model, validation, type = "response")
prediction <- factor(prob > 0.5, levels = c(FALSE, TRUE), labels = c("No", "Yes"))
logit.perf <- table(validation$default, prediction, dnn = c("actual", "predicted"))
logit.perf
```

```
##       predicted
## actual   No  Yes
##    No  2892    7
##    Yes   67   34
```

```
mean(validation$default != prediction)
```

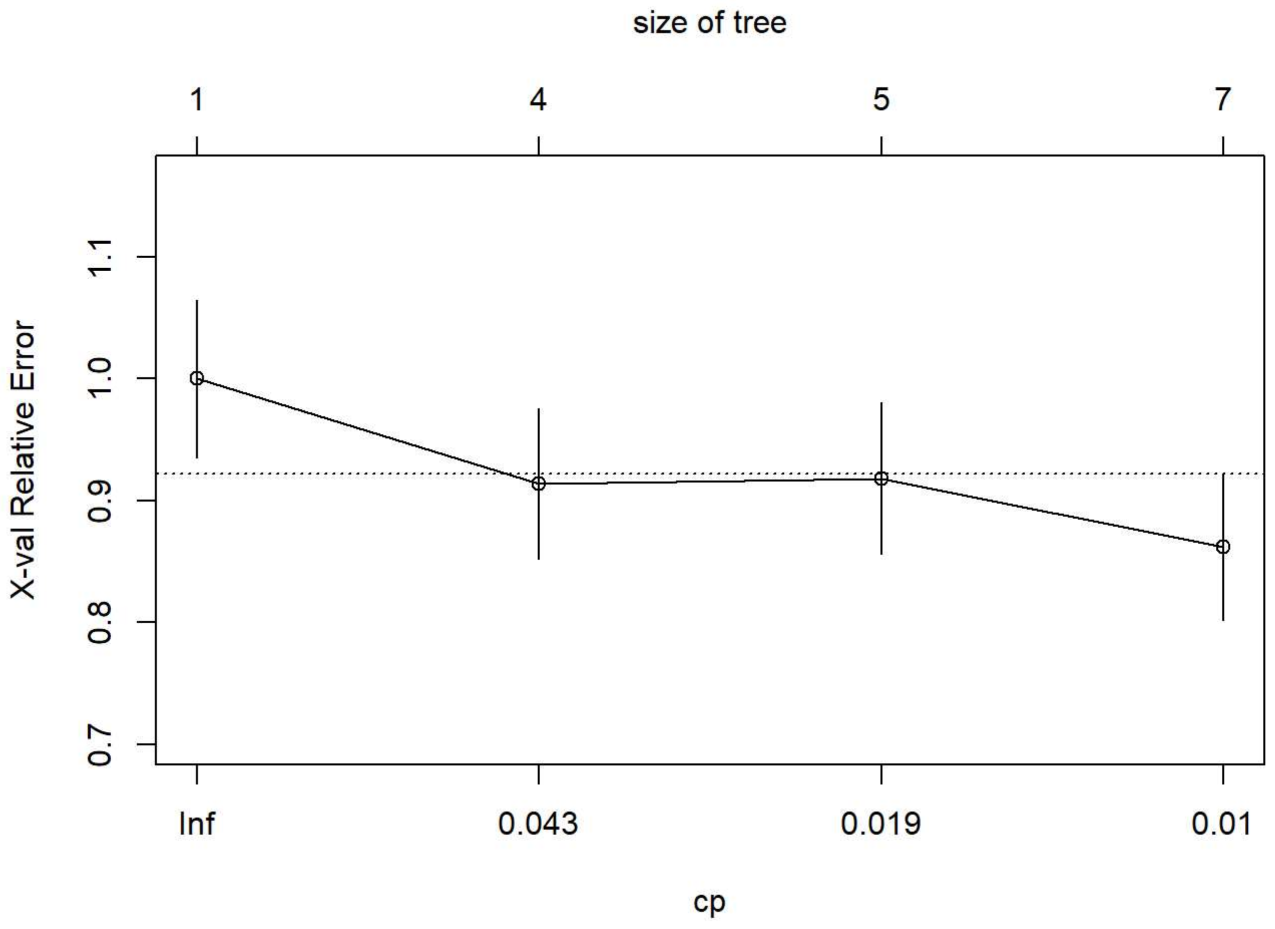
```
## [1] 0.02466667
```

(b)

```
library(rpart)
set.seed(1234)
tree <- rpart(default ~ student + balance + income, data = training, method = "class", parms=li
st(split="information"))
tree$cptable
```

```
##           CP nsplit rel error    xerror       xstd
## 1 0.05459770      0 1.0000000 1.0000000 0.06455608
## 2 0.03448276      3 0.8362069 0.9137931 0.06180190
## 3 0.01077586      4 0.8017241 0.9181034 0.06194293
## 4 0.01000000      6 0.7801724 0.8620690 0.06008035
```

```
plotcp(tree)
```

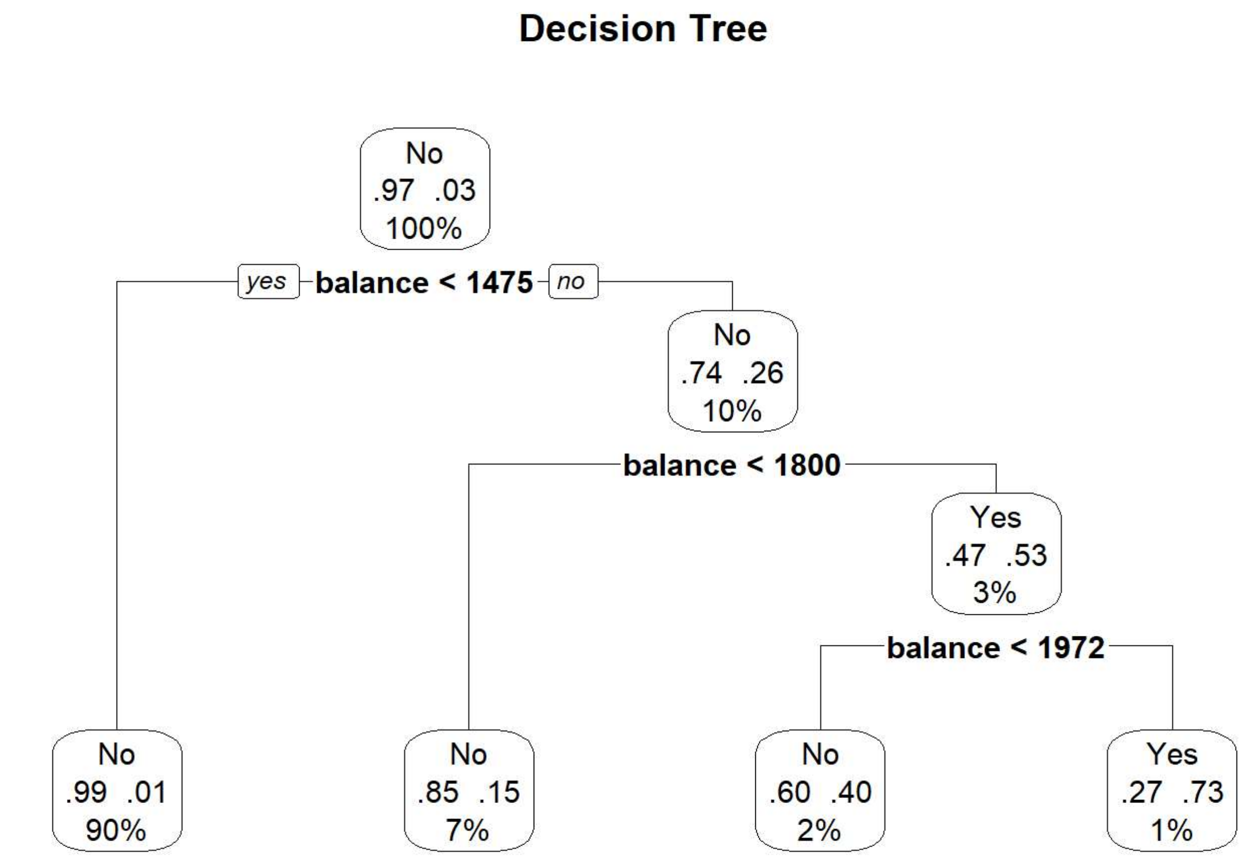


According to the one standard error principle, we should choose the tree whose size is 3.

```
pruned <- prune(tree, cp = 0.04)
library(rpart.plot)
```

```
## Warning: 程辑包'rpart.plot'是用R版本4.3.2 来建造的
```

```
prp(pruned, type = 2, extra = 104,
      fallen.leaves = TRUE, main="Decision Tree")
```



(c)

```
library(party)
```

```
## Warning: 程辑包'party'是用R版本4.3.2 来建造的
```

```
## 载入需要的程辑包：grid
```

```
## 载入需要的程辑包：mvtnorm
```

```
##
## 载入程辑包：'mvtnorm'
```

```
## The following object is masked from 'package:mclust':
##
##     dmvnorm
```

```
## 载入需要的程辑包：modeltools
## 载入需要的程辑包：stats4
```

```
## 载入需要的程辑包：strucchange
```

```
## Warning: 程辑包'strucchange'是用R版本4.3.2 来建造的
```

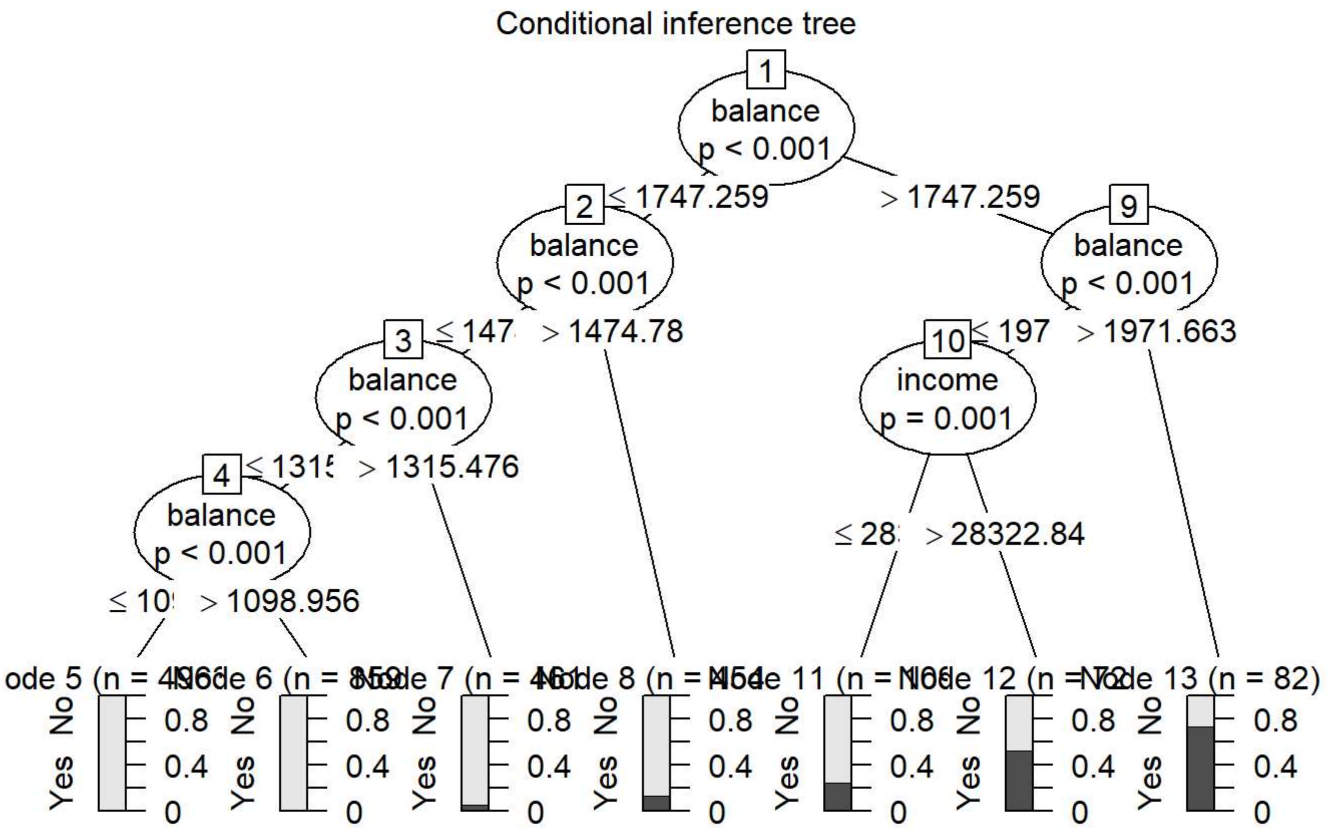
```
## 载入需要的程辑包：zoo
```

```
##
## 载入程辑包：'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## 载入需要的程辑包：sandwich
```

```
set.seed(12)
ctree <- ctree(default ~ student + balance + income, data = training)
plot(ctree, main = "Conditional inference tree")
```



```
pred <- predict(ctree, validation, type = "response")
inference.perf <- table(validation$default, pred, dnn = c("Actual", "Predicted"))
inference.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2883   16
##    Yes   60   41
```

```
mean(validation$default != pred)
```

```
## [1] 0.02533333
```

(d)

Assume that N is the number of cases in the training sample and M is the number of variables. Then the algorithm is as follows:

- Grow a large number of decision trees by sampling N cases with replacement from the training set.
- Sample $m < M$ variables at each node. These variables are considered candidates for splitting in that node. The value m is the same for each node.
- Grow each tree fully without pruning (the minimum node size is set to 1).
- Terminal nodes are assigned to a class based on the mode of cases in that node.
- Classify new cases by sending them down all the trees and taking a vote—majority rules.

Random forest based on decision tree.

```
library(randomForest)
```

```
## Warning: 程辑包'randomForest'是用R版本4.3.2 来建造的
```

```
## randomForest 4.7-1.1
```

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

```
set.seed(12345)
decision.forest <- randomForest(default ~ student + balance + income, data = training, ntree =
500)
decision.pred <- predict(decision.forest, validation, type = "response")
decision.forest.perf <- table(validation$default, decision.pred, dnn = c("Actual", "Predicte
d"))
decision.forest.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2891    8
##    Yes   76   25
```

```
mean(validation$default != decision.pred)
```

```
## [1] 0.028
```

Random forest based on conditional inference tree.

```
library(party)
set.seed(12346)
inference.forest <- cforest(default ~ student + balance + income, data = training, controls = c
forest_unbiased(ntree = 50))
inference.pred <- predict(inference.forest, newdata = validation, OOB = FALSE, type = "respons
e")
inference.forest.perf <- table(validation$default, inference.pred, dnn = c("Actual", "Predicte
d"))
inference.forest.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2887   12
##    Yes   60   41
```

```
mean(validation$default != inference.pred)
```

```
## [1] 0.024
```

We can figure out that the misclassification rate of random forest based on decision tree and conditional inference tree are respectively 0.028 and 0.024. Thus, the accuracy rate of the random forest based on conditional inference tree is higher, which means it performs better than the other one.

(e)

```
library(e1071)
set.seed(1245)
svm.model <- svm(default ~ student + balance + income, data = training, gamma = 1, cost = 1)

svm.pred <- predict(svm.model, newdata = validation)
svm.perf <- table(validation$default, svm.pred, dnn = c("Actual", "Predicted"))
svm.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2894    5
##    Yes   76   25
```

Now campare the performance of different methods.

```
performance <- function(table, n=2){
  if(!all(dim(table) == c(2,2)))
    stop("Must be a 2 x 2 table")
  tn = table[1,1]
  fp = table[1,2]
  fn = table[2,1]
  tp = table[2,2]
  sensitivity = tp/(tp+fn)
  specificity = tn/(tn+fp)
  ppp = tp/(tp+fp)
  npp = tn/(tn+fn)
  hitrate = (tp+tn)/(tp+tn+fp+fn)
  result <- paste("Sensitivity = ", round(sensitivity, n) ,
                  "\nSpecificity = ", round(specificity, n),
                  "\nPositive Predictive Power = ", round(ppp, n),
                  "\nNegative Predictive Power = ", round(npp, n),
                  "\nAccuracy = ", round(hitrate, n), "\n", sep="")
  cat(result)
}

effect <- function(table, n= 2){
  if(!all(dim(table) == c(2,2)))
    stop("Must be a 2 x 2 table")
  tn = table[1,1]
  fp = table[1,2]
  fn = table[2,1]
  tp = table[2,2]
  sensitivity = tp/(tp+fn)
  specificity = tn/(tn+fp)
  ppp = tp/(tp+fp)
  npp = tn/(tn+fn)
  hitrate = (tp+tn)/(tp+tn+fp+fn)
  return(c(sensitivity,specificity,ppp,npp,hitrate))
}
```

SVM

```
performance(svm.perf)
```

```
## Sensitivity = 0.25
## Specificity = 1
## Positive Predictive Power = 0.83
## Negative Predictive Power = 0.97
## Accuracy = 0.97
```

Conditional inference tree

```
performance(inference.perf)
```

```
## Sensitivity = 0.41
## Specificity = 0.99
## Positive Predictive Power = 0.72
## Negative Predictive Power = 0.98
## Accuracy = 0.97
```

Random forest based on decision tree

```
performance(decision.forest.perf)
```

```
## Sensitivity = 0.25
## Specificity = 1
## Positive Predictive Power = 0.76
## Negative Predictive Power = 0.97
## Accuracy = 0.97
```

Random forest based on conditional inference tree

```
performance(inference.forest.perf)
```

```
## Sensitivity = 0.41
## Specificity = 1
## Positive Predictive Power = 0.77
## Negative Predictive Power = 0.98
## Accuracy = 0.98
```

Logistic regression

```
performance(logit.perf)
```

```
## Sensitivity = 0.34
## Specificity = 1
## Positive Predictive Power = 0.83
## Negative Predictive Power = 0.98
## Accuracy = 0.98
```

Now, we put all their performance together to compare, where “ctree” denotes for conditional inference tree, “random_forest” denotes for random forest based on decision tree, “cforest” denotes for random forest based on conditional inference tree and “logit” denotes for logistic regression model.

```
compare <- data.frame( svm = effect(svm.perf), ctree = effect(inference.perf), random_forest =
effect(decision.forest.perf), cforest = effect(inference.forest.perf), logit = effect(logit.per
f))

rownames(compare) <- c("sensitivity","specificity","positive predictive power","negative predic
tive power", "accuracy")

compare
```

```
##                                 svm     ctree random_forest   cforest     logit
## sensitivity               0.2475248 0.4059406     0.2475248 0.4059406 0.3366337
## specificity               0.9982753 0.9944809     0.9972404 0.9958606 0.9975854
## positive predictive power 0.8333333 0.7192982     0.7575758 0.7735849 0.8292683
## negative predictive power 0.9744108 0.9796126     0.9743849 0.9796403 0.9773572
## accuracy                  0.9730000 0.9746667     0.9720000 0.9760000 0.9753333
```