Statistical Calculation and Software

Assignment 4

Hanbin Liu 11912410

4.1

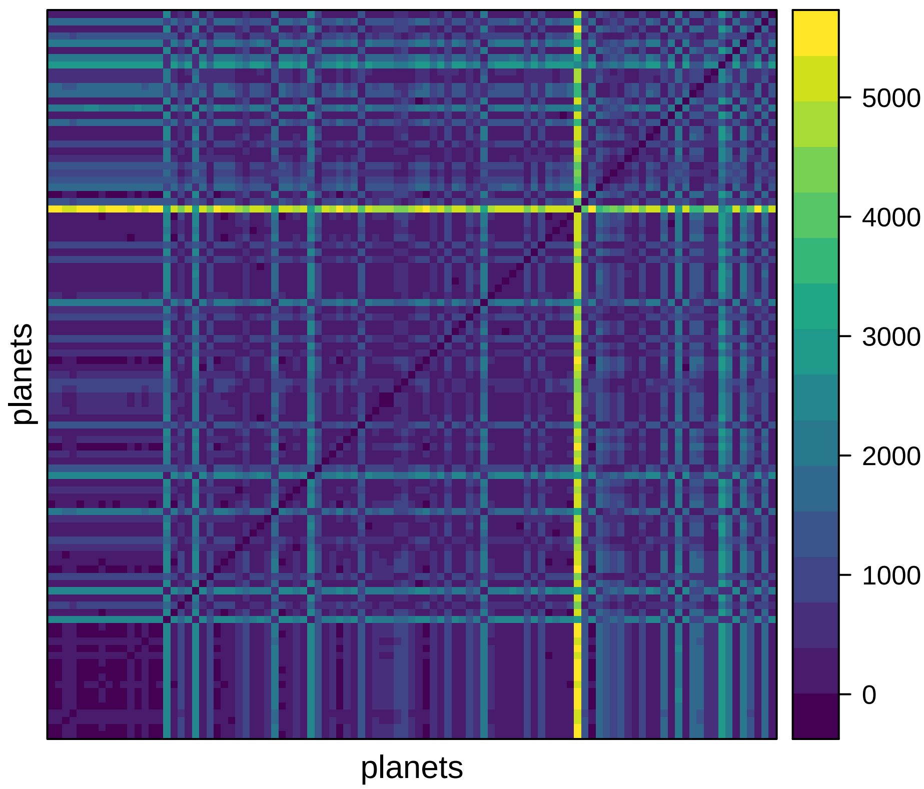
```
library(HSAUR3)
```

```
##      tools
```

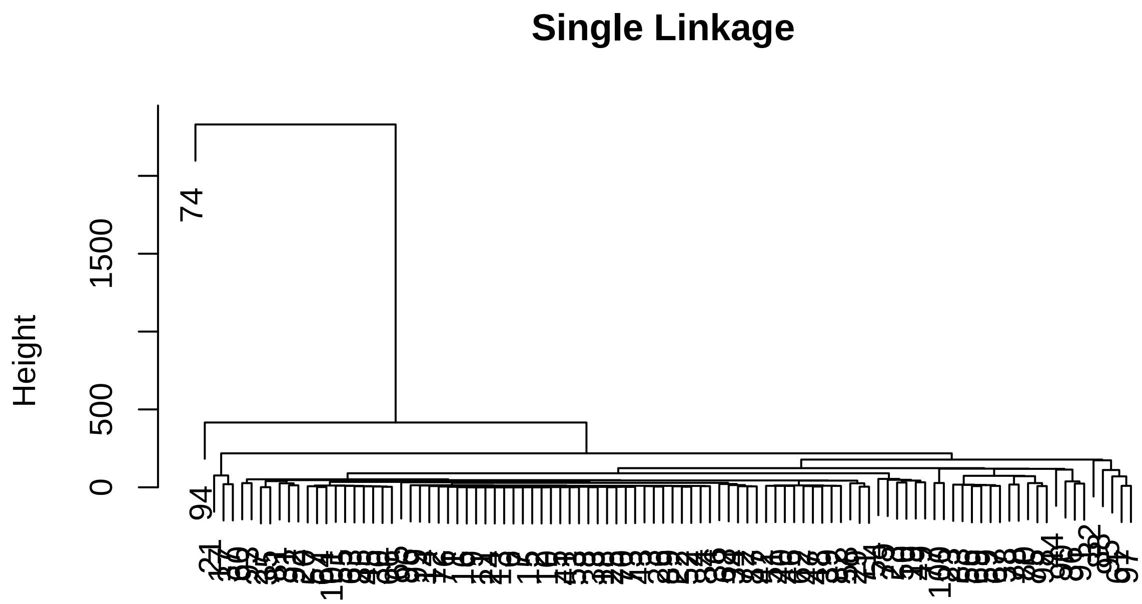
```
data(planets)
```

(a)

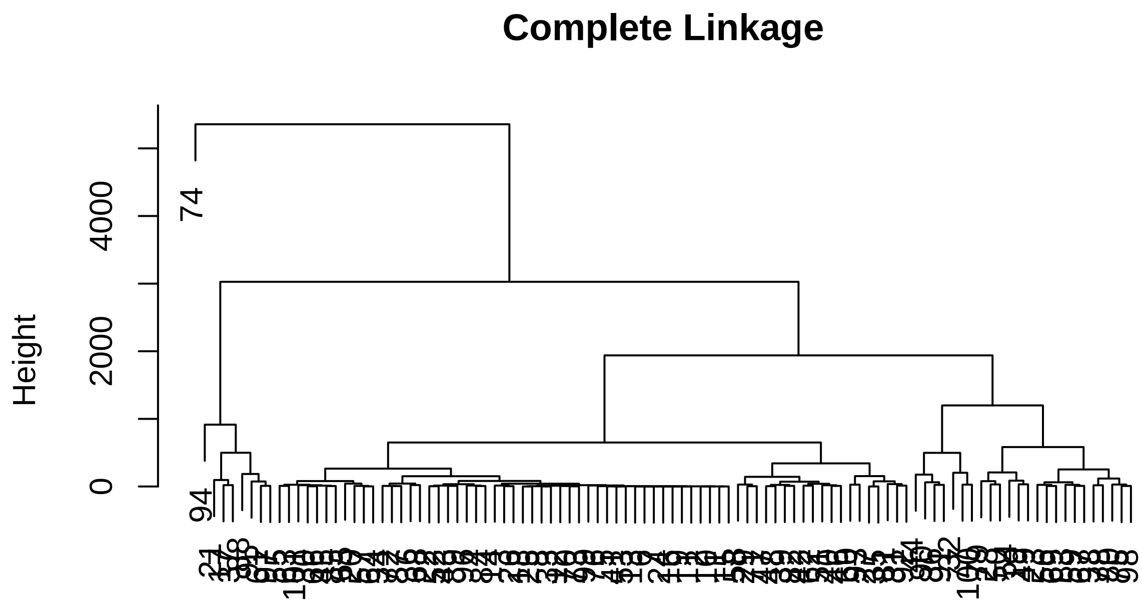
```
library(lattice)
library(viridisLite)
planets_dist <- dist(planets)
levelplot(
  as.matrix(planets_dist),
  xlab = "planets",
  ylab = "planets",
  col.regions = viridis(100),
  scales = list(draw = FALSE)
)
```



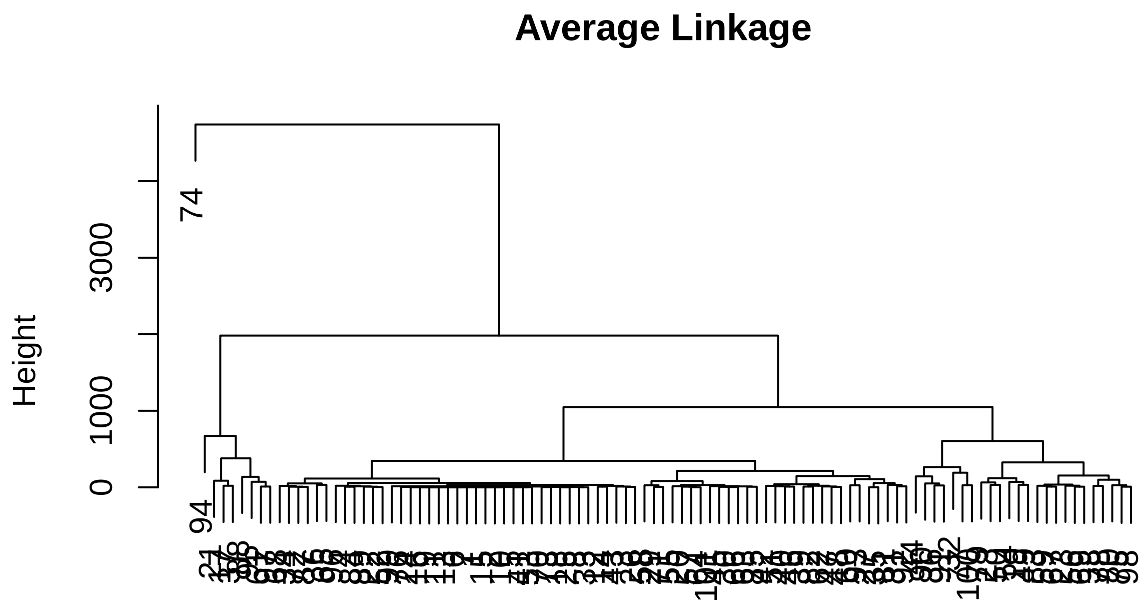

```
planets_single <- hclust(planets_dist, method = "single")
planets_complete <- hclust(planets_dist, method = "complete")
planets_average <- hclust(planets_dist, method = "average")

plot(planets_single,
     main = "Single Linkage",
     sub = "",
     xlab = "",
)
```




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




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




```r
planets_cluster_single <- cutree(planets_single, h = 1000)
planets_cluster_complete <- cutree(planets_complete, h = 1500)
planets_cluster_average <- cutree(planets_average, h = 900)
planets_cluster_single
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   1   1   1   1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1
## 101
##   1
```

```r
planets_cluster_complete
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   3   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   1   3   1   1   1   1   3   1   1   1   3   1   1   1   1   2   3   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
```

```
##   1   1   1   3   1   1   1   1   3   3   1   1   1   3   1   3   1   1   3   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   1   1   1   1   1   3   1   3   1   1   1   1   4   3   1   3   3   3   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   3   1   1   3   1   2   1   3   3   1   1   2   2   1   2   3   1   3
## 101
##   1
```

```r
planets_cluster_average
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   2   1   3   1
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   1   3   1   1   1   1   3   1   1   1   3   1   1   1   1   2   3   1   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   1   1   3   1   1   1   1   3   3   1   1   1   3   1   3   1   1   3   1
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   1   1   1   1   1   3   1   3   1   1   1   1   4   3   1   3   3   3   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   1   3   1   1   3   1   2   1   3   3   1   1   2   2   1   2   3   1   3
## 101
##   1
```

(b)

```r
## min-max standardized
rge <- apply(planets, 2, max) - apply(planets, 2, min)
planet.dat <-
  sweep(planets, 2, rge, FUN = "/") ### function = divide

## K=3
planet_kmeans3 <- kmeans(planet.dat, centers = 3)
planet_kmeans3
```

```
## K-means clustering with 3 clusters of sizes 14, 53, 34
##
## Cluster means:
##         mass     period     eccen
## 1 0.60560786 0.31606632 0.3953614
## 2 0.09576256 0.07984122 0.1315524
## 3 0.16777347 0.11500361 0.5343613
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   2   3   2   2   2   2   3   2   2   2   2   2   2   2   2   2   2   2   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   3   2   3   2   2   2   2   2   3   3   3   3   2   3   2   2   2   2   3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   2   2   3   2   2   3   3   3   3   2   2   2   2   3   2   2   2   2   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   2   3   3   2   2   3   3   2   3   3   2   1   3   2   3   3   3   3
```

```
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100 
##   3   2   2   2   3   1   3   1   3   1   1   3   1   1   1   1   1   1   1   1 
## 101 
##   1 
## 
## Within cluster sum of squares by cluster:
## [1] 1.719130 1.755694 1.787017
##  (between_SS / total_SS =  57.2 %)
## 
## Available components:
## 
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(planet_kmeans3$cluster)
```

```
## 
##  1  2  3 
## 14 53 34
```

```
ccent <- function(cl) {
  f <- function(i)
    colMeans(planets[cl == i, ])
  x <- sapply(sort(unique(cl)), f)
  colnames(x) <- sort(unique(cl))
  return(x)
}

ccent(planet_kmeans3$cluster)
```

```
##                 1           2           3
## mass     10.56786   1.6710566   2.9276471
## period 1693.17201 427.7105892 616.0760882
## eccen     0.36650   0.1219491   0.4953529
```

```
## K=5
planet_kmeans5 <- kmeans(planet.dat, centers = 5)
planet_kmeans5
```

```
## K-means clustering with 5 clusters of sizes 14, 17, 30, 8, 32
## 
## Cluster means:
##         mass     period     eccen
## 1 0.61960704 0.24615398 0.4138542
## 2 0.21051744 0.12598649 0.6574656
## 3 0.09991595 0.03290963 0.0531931
## 4 0.11840974 0.44869904 0.2067152
## 5 0.09563037 0.07505714 0.3267260
## 
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20 
##   3   3   5   5   3   3   3   5   3   3   3   5   3   5   3   3   4   5   3   2
```

```
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   3   5   3   3   3   3   5   5   5   2   4   3   5   5   5   4   4   5   5
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   2   4   5   5   2   5   2   5   5   3   3   2   5   5   5   5   3   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   4   5   5   5   5   5   5   2   2   3   5   2   3   4   2   3   4   5   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   3   3   3   2   1   1   1   2   1   1   2   1   1   1   1   1   1   1   1
## 101
##   1
##
## Within cluster sum of squares by cluster:
## [1] 1.1038061 0.7801019 0.4920874 0.5852326 0.5153905
##  (between_SS / total_SS =  71.7 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```
table(planet_kmeans5$cluster)
```

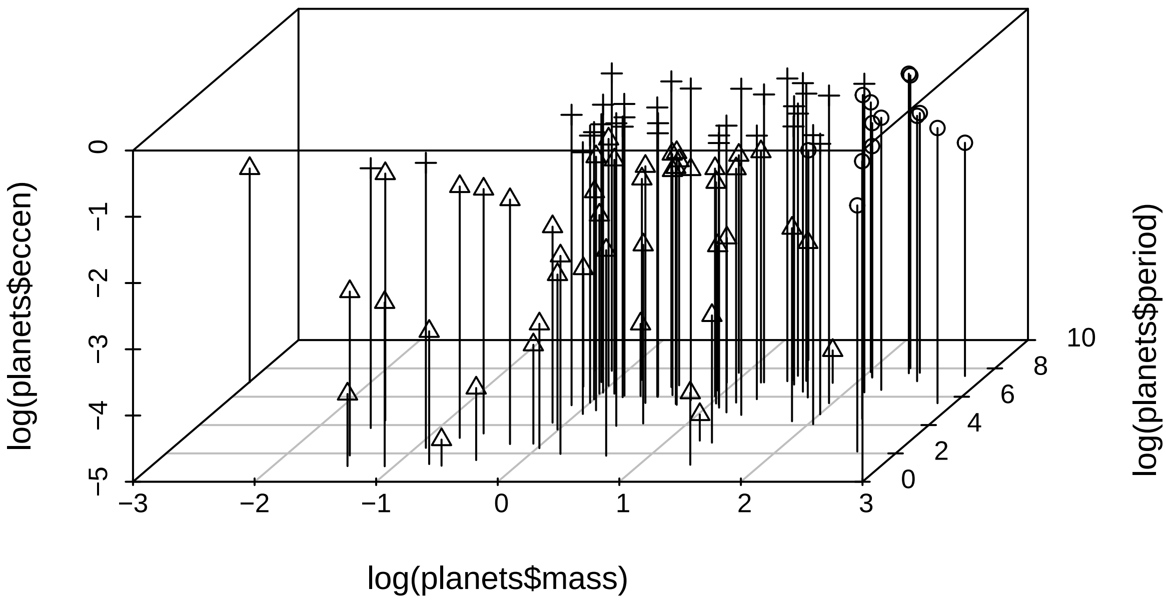
```
##
##  1  2  3  4  5
## 14 17 30  8 32
```

```
ccent(planet_kmeans5$cluster)
```

```
##                   1           2          3           4          5
## mass      10.8121429   3.6735294   1.743533    2.066250   1.668750
## period 1318.6505856 674.9115294 176.297374 2403.687500 402.082219
## eccen      0.3836429   0.6094706   0.049310    0.191625   0.302875
```

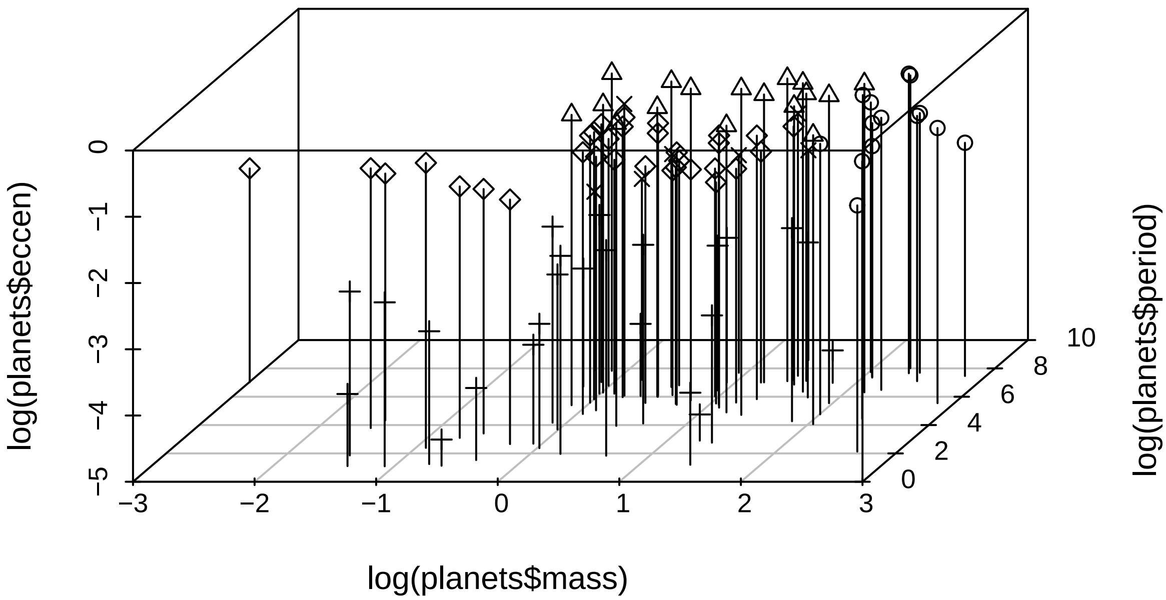
- 3D scatterplot for K=3

```
library("scatterplot3d")
layout(matrix(1))
scatterplot3d(
  log(planets$mass),
  log(planets$period),
  log(planets$eccen),
  type = "h",
  angle = 55,
  scale.y = 0.7,
  pch = planet_kmeans3$cluster,
  y.ticklabs = seq(0, 10, by = 2),
  y.margin.add = 0.1,
)
```



- 3D scatterplot for K=5

```
library("scatterplot3d")
layout(matrix(1))
scatterplot3d(
  log(planets$mass),
  log(planets$period),
  log(planets$eccen),
  type = "h",
  angle = 55,
  scale.y = 0.7,
  pch = planet_kmeans5$cluster,
  y.ticklabs = seq(0, 10, by = 2),
  y.margin.add = 0.1,
)
```



(c)

```
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))
}
x <- planets$eccen
startparam <-
  c(
    p = 0.5,
    mu1 = mean(x) / 2,
    sd1 = sd(x) / 2,
    mu2 = mean(x) * 2,
    sd2 = sd(x) * 2
  )

opp <-
  optim(
    startparam,
    logL,
    x = planets$eccen,
    method = "L-BFGS-B",
    lower = c(0.01, rep(0.01, 4)),
```

```r
    upper = c(0.99, rep(1, 4))
  )

opp
```

```
## $par
##          p        mu1        sd1        mu2        sd2
## 0.18979358 0.02445790 0.02447048 0.34176842 0.18742605
##
## $value
## [1] -27.81359
##
## $counts
## function gradient
##       59       59
##
## $convergence
## [1] 0
##
## $message
## [1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

(d)

```r
library("mclust")
```

```
## Package 'mclust' version 5.4.7
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
planet_mclust <- Mclust(planet.dat[3], G = 2)
print(planet_mclust)
```

```
## 'Mclust' model object: (V,2)
##
## Available components:
##  [1] "call"           "data"           "modelName"      "n"
##  [5] "d"              "G"              "BIC"            "loglik"
##  [9] "df"             "bic"            "icl"            "hypvol"
## [13] "parameters"     "z"              "classification" "uncertainty"
```

```r
table(planet_mclust$classification)
```

```
##
##  1  2
## 21 80
```

```r
## sample statistics
ind1 <- planet_mclust$classification == 1
ind2 <- planet_mclust$classification == 2
data.frame(mclust.BIC = c(
  p = length(x[ind1]) / length(x),
  mu1 = mean(x[ind1]),
  sd1 = sd(x[ind1]),
  mu2 = mean(x[ind2]),
  sd2 = sd(x[ind2])
))
```

```
##     mclust.BIC
## p   0.20792079
## mu1 0.02100000
## sd1 0.02110924
## mu2 0.34994125
## sd2 0.18283981
```

The results match the results in (c). Compare:

```r
data.frame(loglikelihood = opp$par,
           mclust.BIC = c(21 / 101, mean(x[ind1]), sd(x[ind1]), mean(x[ind2]), sd(x[ind2])))
```

```
##     loglikelihood mclust.BIC
## p      0.18979358 0.20792079
## mu1    0.02445790 0.02100000
## sd1    0.02447048 0.02110924
## mu2    0.34176842 0.34994125
## sd2    0.18742605 0.18283981
```

(e)

```r
planets_pca <- prcomp(planets, scale = TRUE)
```

coefficients for the first two principal components:

```r
planets_pca$rotation[, 1]
```

```
##      mass    period     eccen
## 0.6423065 0.5229950 0.5602843
```

```r
planets_pca$rotation[, 2]
```

```
##        mass      period       eccen
## -0.05996314  0.76306298 -0.64353657
```

score:

```
first.score <- predict(planets_pca)[, 1]
second.score <- predict(planets_pca)[, 2]
third.score <- predict(planets_pca)[, 3]

score <- data.frame(first = first.score,
                    second = second.score,
                    third = third.score)
score
```

```
##           first       second        third
## 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)

```r
## min-max standardized
rge <- apply(score, 2, max) - apply(score, 2, min)
score.dat <-
  sweep(score, 2, rge, FUN = "/") ### function = divide

## K=3
score_kmeans3 <- kmeans(score.dat[c(1, 2)], centers = 3)
```

compare:

```r
score_kmeans3
```

```
## K-means clustering with 3 clusters of sizes 28, 6, 67
##
## Cluster means:
##        first      second
## 1  0.2285479 -0.08306881
## 2  0.5393149  0.20697410
## 3 -0.1438094  0.01618033
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   1   3   3   3   3   3   3   3   3   3   1   1   3   3   3   3   3   3   3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   3   3   1   3   3   3   1   3   1   3   3   3   3   1   3   3   3   3   3   3
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   3   3   3   3   3   3   3   1   1   3   3   1   3   2   1   3   1   1   1   1
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   1   3   3   3   1   1   1   2   1   1   1   1   1   2   2   1   2   1   1   2
## 101
##   1
##
## Within cluster sum of squares by cluster:
## [1] 0.8045553 0.2845129 1.5241556
##  (between_SS / total_SS =  65.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
planet_kmeans3
```

```
## K-means clustering with 3 clusters of sizes 14, 53, 34
##
## Cluster means:
##         mass     period     eccen
## 1 0.60560786 0.31606632 0.3953614
## 2 0.09576256 0.07984122 0.1315524
## 3 0.16777347 0.11500361 0.5343613
##
## Clustering vector:
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   2   3   2   2   2   2   3   2   2   2   2   2   2   2   2   2   2   2   3
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   3   2   3   2   2   2   2   2   3   3   3   3   2   3   2   2   2   2   3   3
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   2   2   3   2   2   3   3   3   3   2   2   2   2   3   2   2   2   2   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   2   2   3   3   2   2   3   3   2   3   3   2   1   3   2   3   3   3   3
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   3   2   2   2   3   1   3   1   3   1   1   3   1   1   1   1   1   1   1   1
## 101
##   1
##
## Within cluster sum of squares by cluster:
## [1] 1.719130 1.755694 1.787017
##  (between_SS / total_SS =  57.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

```r
table(score_kmeans3$cluster)
```

```
##
##  1  2  3
## 28  6 67
```

```r
table(planet_kmeans3$cluster)
```

```
##
##  1  2  3
## 14 53 34
```

The results are significantly different from the results in (b).

4.2

```r
library(ISLR)
data(Default)
```

(a)

```r
## split the sample set
set.seed(1234)
train <- sample(nrow(Default), 0.7 * nrow(Default))
Default.train <- Default[train, ]
Default.validate <- Default[-train, ]

table(Default.train$default)
```

```
##
##   No  Yes
## 6769  231
```

```r
table(Default.validate$default)
```

```
##
##   No  Yes
## 2898  102
```

```r
## logistic regression
fit.logit <-
  glm(default ~ student + balance + income,
      data = Default.train,
      family = binomial())
summary(fit.logit)
```

```
##
## Call:
## glm(formula = default ~ student + balance + income, family = binomial(),
##     data = Default.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1410  -0.1435  -0.0573  -0.0208   3.6583
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.064e+01  5.867e-01 -18.129  < 2e-16 ***
## studentYes  -7.078e-01  2.747e-01  -2.577  0.00998 **
## balance      5.666e-03  2.752e-04  20.586  < 2e-16 ***
## income      -4.095e-07  9.829e-06  -0.042  0.96677
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
```

```
##
##     Null deviance: 2030.3  on 6999  degrees of freedom
## Residual deviance: 1111.6  on 6996  degrees of freedom
## AIC: 1119.6
##
## Number of Fisher Scoring iterations: 8
```

```
prob <- predict(fit.logit, Default.validate, type = "response")

logit.pred <- factor(prob > .5,
                     levels = c(FALSE, TRUE),
                     labels = c("No", "Yes"))
logit.perf <- table(Default.validate$default,
                    logit.pred,
                    dnn = c("Actual", "Predicted"))

## confusion matrix
logit.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2887   11
##    Yes   65   37
```

```
## validation set error
error <-
  (logit.perf[1, 2] + logit.perf[2, 1]) / (nrow(Default.validate))
error
```

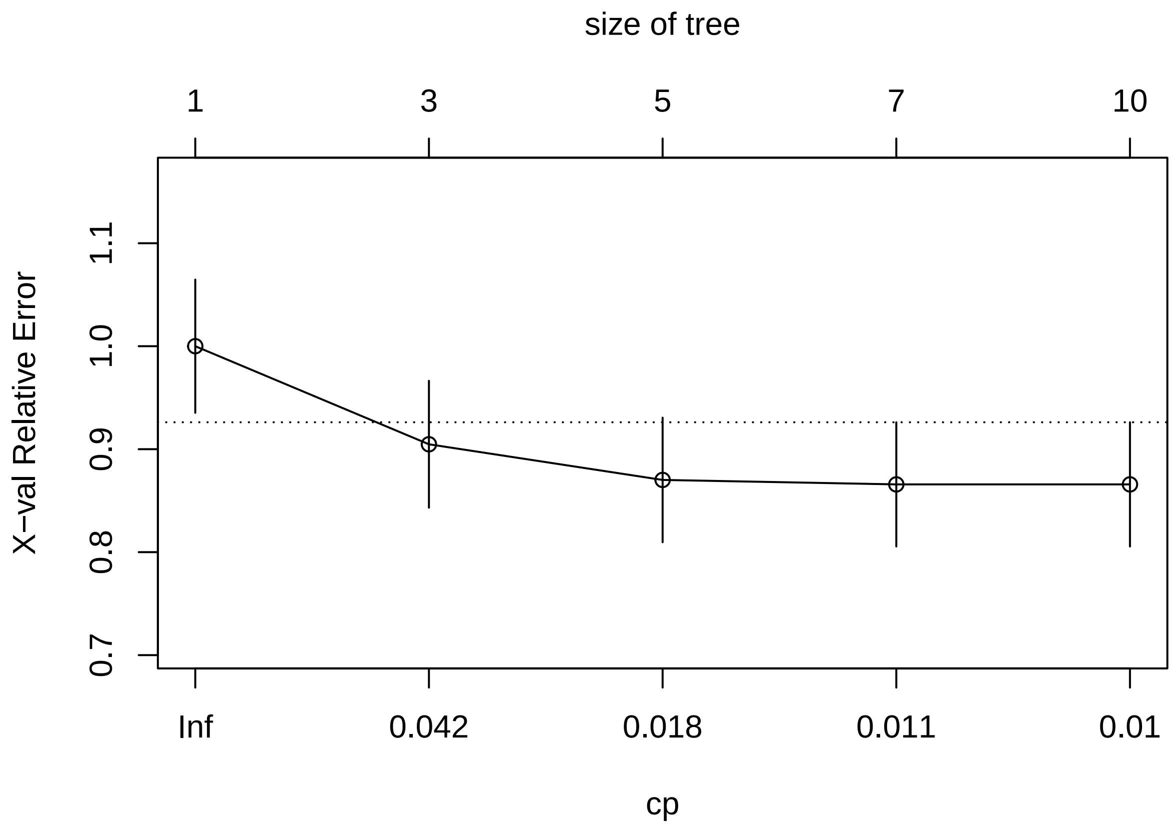
```
## [1] 0.02533333
```

(b)

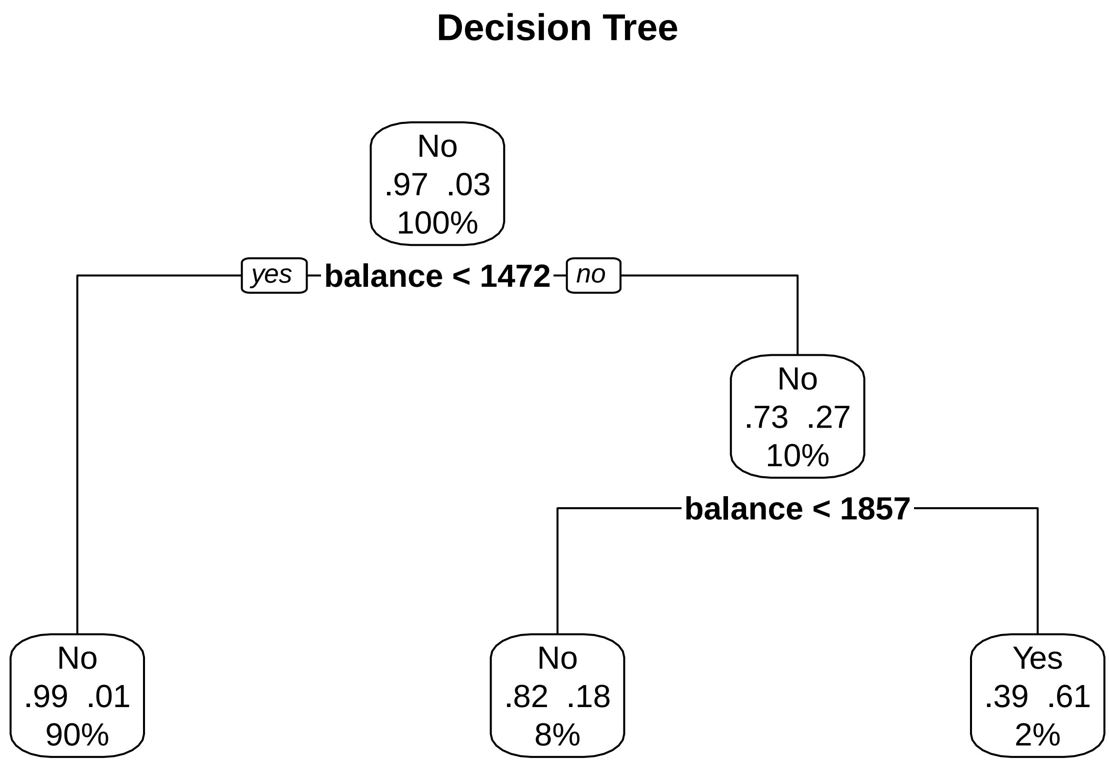
```
library(rpart)
set.seed(1234)

dtree <- rpart(
  default ~ student + balance + income,
  data = Default.train,
  method = "class",
  parms = list(split = "information")
)

dtree$cptable
```

```
##           CP nsplit rel error    xerror       xstd
## 1 0.06926407      0 1.0000000 1.0000000 0.06470044
## 2 0.02597403      2 0.8614719 0.9047619 0.06164232
## 3 0.01298701      4 0.8095238 0.8701299 0.06048665
## 4 0.01010101      6 0.7835498 0.8658009 0.06034044
## 5 0.01000000      9 0.7532468 0.8658009 0.06034044
```

```
plotcp(dtree)
```




```
dtree.pruned <- prune(dtree, cp = 0.039)
library(rpart.plot)
prp(
  dtree.pruned,
  type = 2,
  extra = 104,
  fallen.leaves = TRUE,
  main = "Decision Tree"
)
```



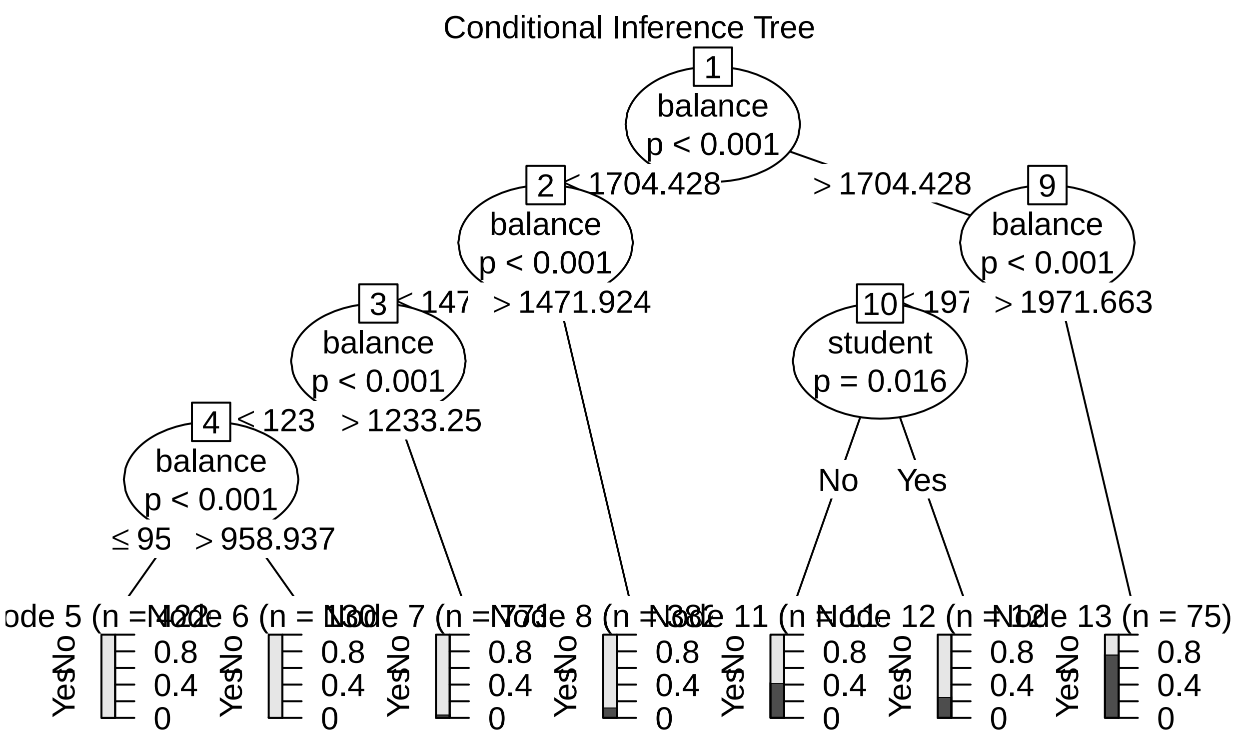
(C)

```
library(party)
```

```
##      grid

##      mvtnorm

##
##    'mvtnorm'

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

##      modeltools

##      stats4

##      strucchange

##      zoo
```

```
##
##     'zoo'

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

##      sandwich
```

```r
fit.ctree <- ctree(default ~ student + balance + income,
                   data = Default.train)
plot(fit.ctree, main = "Conditional Inference Tree")
```




```r
## prediction in the validation set
ctree.pred <-
  predict(fit.ctree, Default.validate, type = "response")

ctree.perf <- table(Default.validate$default,
                    ctree.pred,
                    dnn = c("Actual", "Predicted"))
## confusion matrix
ctree.perf
```

```
##       Predicted
```

```
## Actual   No  Yes
##    No  2888   10
##    Yes   69   33
```

(d)

- traditional decision trees:

```r
library(randomForest)
```

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

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

```r
set.seed(1234)

## grow the forest
fit.forest1 <-
  randomForest(
    default ~ student + balance + income,
    data = Default.train,
    na.action = na.roughfix,
    importance = TRUE
  )
fit.forest1
```

```
##
## Call:
##  randomForest(formula = default ~ student + balance + income,      data = Default.train, importance =
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 1
##
##         OOB estimate of  error rate: 2.96%
## Confusion matrix:
##       No Yes class.error
## No   6752  17 0.002511449
## Yes   190  41 0.822510823
```

```r
forest1.pred <- predict(fit.forest1, Default.validate)
forest1.perf <- table(Default.validate$default,
                     forest1.pred,
                     dnn = c("Actual", "Predicted"))
## confusion matrix
forest1.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2889    9
##    Yes   76   26
```

- conditional inference trees:

```
## grow the forest
fit.forest2 <-
  cforest(default ~ student + balance + income,
          data = Default.train,
          controls = cforest_classical(mtry = 2))
fit.forest2
```

```
##
##   Random Forest using Conditional Inference Trees
##
## Number of trees:  500
##
## Response:  default
## Inputs:  student, balance, income
## Number of observations:  7000
```

```
forest2.pred <- predict(fit.forest2, newdata = Default.validate)
forest2.perf <- table(Default.validate$default,
                      forest2.pred,
                      dnn = c("Actual", "Predicted"))
## confusion matrix
forest2.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2887   11
##    Yes   65   37
```

Compare the predictive accuracy

```
forest.traditional.accuracy  <-
  (forest1.perf[1, 1] + forest1.perf[2, 2]) / nrow(Default.validate)
forest.conditional.accuracy <-
  (forest2.perf[1, 1] + forest2.perf[2, 2]) / nrow(Default.validate)

data.frame(forest.traditional.accuracy, forest.conditional.accuracy)
```

```
##   forest.traditional.accuracy forest.conditional.accuracy
## 1                   0.9716667                   0.9746667
```

The accuracy of random forest based on the conditional inference trees is slightly larger than the accuracy of random forest based on the traditional decision trees.

(e)

```
library(e1071)
set.seed(1234)
fit.svm <-
```

```r
  svm(
    default ~ student + balance + income,
    data = Default.train,
    gamma = 1,
    cost = 1
  )
fit.svm
```

```
##
## Call:
## svm(formula = default ~ student + balance + income, data = Default.train,
##     gamma = 1, cost = 1)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  1
##
## Number of Support Vectors:  616
```

```r
svm.pred <- predict(fit.svm, na.omit(Default.validate))
svm.perf <- table(na.omit(Default.validate)$default,
                  svm.pred,
                  dnn = c("Actual", "Predicted"))
svm.perf
```

```
##       Predicted
## Actual   No  Yes
##    No  2891    7
##    Yes   76   26
```

- Compare:

```r
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 <-
    data.frame(c(sensitivity,
                 specificity,
                 ppp,
                 npp,
                 hitrate))
```

```
  return(result)
}


result <- data.frame(
  svm = performance(svm.perf),
  conditionaltree = performance(ctree.perf),
  forest1 = performance(forest1.perf),
  forest2 = performance(forest2.perf),
  logit = performance(logit.perf),
  row.names = c(
    "sensitivity",
    "specificity",
    "positie predictive power",
    "negative predicive power",
    "accuracy"
  )
)
colnames(result) <-
  c("svm", "cond.tree", "forest.trad", "forest.cond", "logit")
result
```

```
##                                svm cond.tree forest.trad forest.cond     logit
## sensitivity              0.2549020 0.3235294   0.2549020   0.3627451 0.3627451
## specificity              0.9975845 0.9965493   0.9968944   0.9962043 0.9962043
## positie predictive power 0.7878788 0.7674419   0.7428571   0.7708333 0.7708333
## negative predicive power 0.9743849 0.9766655   0.9743676   0.9779810 0.9779810
## accuracy                 0.9723333 0.9736667   0.9716667   0.9746667 0.9746667
```

Conclusion:

Random forest based on the conditional trees has the same performance as the logistic regression. They have the best sensitivity, negative predictive power, and accuracy.

While svm has the best specificity and positive predictive power.