39_TilakPoudel 2nd Assessment

Tilak Poudel

2025-05-28

Q.N 6

Create a dataset with following variables: age (20-59) years height (110-190) cm weight (40-90) kg with random 150 cases of each variable with random seed 39

a

```
working_directory <- getwd()
print(working_directory)
```

```
## [1] "/home/tilak/projects/tilak/mds1/R-programming/exam/2nd-assessment/39_TilakPoudel"
```

```
# Change to the desired directory
# setwd("~/projects/tilak/mds1/R-programming/")
set.seed(39)
# Generate the data
data <- data.frame(
  age = sample(20:59, 150, replace = TRUE),
  height = sample(110:190, 150, replace = TRUE),
  weight = sample(40:90, 150, replace = TRUE)
)

print(data)
```

```
##     age height weight
## 1    38    143     67
## 2    27    126     63
## 3    59    142     68
## 4    33    154     87
## 5    43    115     68
## 6    31    185     76
## 7    32    187     62
## 8    37    129     84
## 9    49    175     48
## 10   29    138     67
## 11   33    178     86
## 12   33    139     82
## 13   43    145     45
```

```
## 14   49    169     64
## 15   52    175     81
## 16   43    169     79
## 17   57    178     67
## 18   46    125     43
## 19   22    140     86
## 20   54    155     72
## 21   59    130     58
## 22   54    148     58
## 23   51    130     43
## 24   37    179     83
## 25   50    183     82
## 26   21    120     87
## 27   25    130     63
## 28   54    171     63
## 29   52    160     68
## 30   21    115     73
## 31   53    190     44
## 32   30    164     79
## 33   28    186     42
## 34   29    190     42
## 35   44    124     81
## 36   31    176     74
## 37   35    130     56
## 38   44    127     72
## 39   23    142     84
## 40   59    131     50
## 41   38    167     72
## 42   30    176     60
## 43   54    153     64
## 44   56    173     67
## 45   34    151     77
## 46   50    115     47
## 47   56    144     63
## 48   47    113     43
## 49   40    165     41
## 50   59    127     66
## 51   36    126     60
## 52   38    120     62
## 53   59    172     75
## 54   32    159     81
## 55   34    148     72
## 56   36    152     90
## 57   45    171     43
## 58   29    170     85
## 59   54    118     54
## 60   23    153     59
## 61   40    185     63
## 62   58    119     90
## 63   24    160     76
## 64   46    129     74
## 65   30    177     90
## 66   59    181     51
## 67   46    113     62
```

```
## 68   31    125     86
## 69   59    118     78
## 70   38    130     41
## 71   45    122     40
## 72   53    113     59
## 73   44    121     50
## 74   55    116     74
## 75   23    182     50
## 76   52    116     72
## 77   55    182     52
## 78   49    125     73
## 79   26    111     84
## 80   31    176     88
## 81   42    178     65
## 82   31    113     48
## 83   26    187     48
## 84   57    132     75
## 85   24    190     69
## 86   57    130     49
## 87   46    151     64
## 88   26    187     54
## 89   20    155     55
## 90   58    155     90
## 91   33    169     42
## 92   24    126     83
## 93   41    143     80
## 94   23    154     74
## 95   26    169     75
## 96   28    172     52
## 97   27    141     90
## 98   30    185     89
## 99   43    152     77
## 100  30    171     53
## 101  42    164     69
## 102  27    137     58
## 103  47    117     89
## 104  54    154     66
## 105  42    183     55
## 106  27    181     69
## 107  28    147     79
## 108  36    136     74
## 109  33    143     48
## 110  22    137     55
## 111  33    169     75
## 112  22    137     88
## 113  58    162     45
## 114  24    149     49
## 115  24    137     65
## 116  22    118     66
## 117  25    123     71
## 118  38    170     76
## 119  29    152     50
## 120  42    185     62
## 121  53    129     72
```

```
## 122  52    144     43
## 123  38    165     71
## 124  48    114     73
## 125  48    122     62
## 126  42    152     64
## 127  32    186     72
## 128  40    134     67
## 129  37    124     64
## 130  58    179     80
## 131  35    148     83
## 132  41    127     55
## 133  54    136     61
## 134  49    126     51
## 135  33    190     52
## 136  31    166     42
## 137  22    170     89
## 138  30    164     84
## 139  38    150     75
## 140  33    118     87
## 141  29    141     69
## 142  50    182     77
## 143  31    172     67
## 144  45    188     79
## 145  42    175     58
## 146  43    177     54
## 147  45    161     48
## 148  49    150     46
## 149  50    158     85
## 150  29    189     45
```

b compute the body mass index variable as : BMI = weight in kg / height in meter squared

```r
data$BMI <- data$weight / (data$height / 100)^2
print(head(data))
```

```
##   age height weight      BMI
## 1  38    143     67 32.76444
## 2  27    126     63 39.68254
## 3  59    142     68 33.72347
## 4  33    154     87 36.68410
## 5  43    115     68 51.41777
## 6  31    185     76 22.20599
```

c create a body mass index category variable as follows: <18, 18-24, 25-30, 30+ and label them as "underweight", "normal", "overweight", "obese" using dplyr package

```
library(dplyr)
```

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

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

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

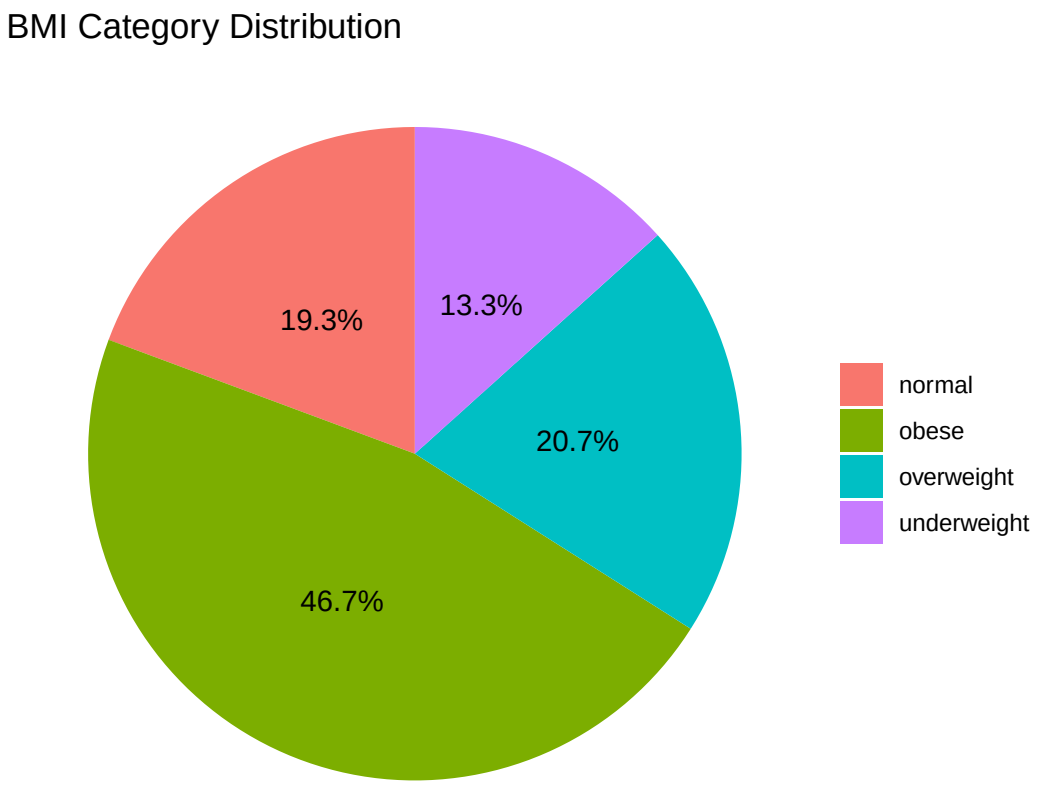
```
data <- data %>%
  mutate(BMI_category = case_when(
    BMI < 18 ~ "underweight",
    BMI >= 18 & BMI < 25 ~ "normal",
    BMI >= 25 & BMI < 30 ~ "overweight",
    TRUE ~ "obese"
  ))
print(head(data))
```

```
##   age height weight      BMI BMI_category
## 1  38    143     67 32.76444        obese
## 2  27    126     63 39.68254        obese
## 3  59    142     68 33.72347        obese
## 4  33    154     87 36.68410        obese
## 5  43    115     68 51.41777        obese
## 6  31    185     76 22.20599       normal
```

show percentage distribution of labelled BMI variable with pie chart using ggplot2 package

```
library(ggplot2)
ggplot(data, aes(x = "", fill = BMI_category)) +
  geom_bar(width = 1) +
  coord_polar(theta = "y") +
  labs(title = "BMI Category Distribution") +
  theme_void() +
  theme(legend.title = element_blank()) +
  geom_text(stat = 'count', aes(label = scales::percent(..count../sum(..count..))), position = position
```

```
## Warning: The dot-dot notation (‘..count..‘) was deprecated in ggplot2 3.4.0.
## i Please use ‘after_stat(count)‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



From the pie chart we can see that the distribution of BMI categories is as follows:

- Underweight: 13.3%
- Normal: 19.3%
- Overweight: 20.7%
- Obese: 46.7%

Q.N 7

Using airquality dataset, do the following:

a. Perform notmality of “Temp” variable by each category of “Month” variable and interpret

```
library(dplyr)
data("airquality")
print(head(airquality))
```

```
##   Ozone Solar.R Wind Temp Month Day
## 1    41     190  7.4   67     5   1
## 2    36     118  8.0   72     5   2
## 3    12     149 12.6   74     5   3
## 4    18     313 11.5   62     5   4
## 5    NA      NA 14.3   56     5   5
## 6    28      NA 14.9   66     5   6
```

```
# see category of Month variable
print(unique(airquality$Month))
```

```
## [1] 5 6 7 8 9
```

```
shapiro_results <- airquality %>%
  group_by(Month) %>%
  summarise(
    shapiro_p_value = shapiro.test(Temp)$p.value
  )
print(shapiro_results)
```

```
## # A tibble: 5 x 2
##   Month shapiro_p_value
##   <int>           <dbl>
## 1     5           0.135
## 2     6           0.583
## 3     7           0.119
## 4     8           0.369
## 5     9           0.183
```

Interpretation

The Shapiro-Wilk test results indicate the following p-values for the "Temp" variable by each month: - May: 0.134 - June: 583 - July: 119 - August: 368 - September: 0.183

For all months, Temp appears to follow a normal distribution (`p > 0.05`), so parametric tests (like t-tests or ANOVA) could be appropriate.

b. Perform test of equality of variance of "Temp" variable by each category of "Month" variable and interpret

```
library(car)
```

```
## Loading required package: carData
```

```
##
## Attaching package: 'car'
```

```
## The following object is masked from 'package:dplyr':
##
##     recode
```

```
# Perform Levene's test for equality of variance
levene_results <- leveneTest(Temp ~ factor(Month), data = airquality)
print(levene_results)
```

```
## Levene's Test for Homogeneity of Variance (center = median)
##        Df F value  Pr(>F)
## group   4  2.5849 0.03941 *
##       148
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation

The Levene's test results indicate the following: - F-value: 2.584 - p-value: 0.0394 This suggests that there is a `significant difference in the variances of the "Temp" variable across different months` (`p < 0.05`). Therefore, we cannot assume equal variances for the "Temp" variable across the months.

c. Compare temp by month using best statistical test for this data and interpret

Since the data is

Data is approximately normal, but

Variances are not equal, we will use ANOVA compare the means of "Temp" across different months.

```
# Perform ANOVA to compare "Temp" across different months
anova_results <- aov(Temp ~ factor(Month), data = airquality)
summary_anova <- summary(anova_results)
print(summary_anova)
```

```
##                Df Sum Sq Mean Sq F value Pr(>F)
## factor(Month)   4   7061  1765.3   39.85 <2e-16 ***
## Residuals     148   6557    44.3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Interpretation

F-value = 39.85, and

p-value < 2e-16, which is much less than 0.05.

So we can say, there is a statistically significant difference in mean temperature across the five months (May to September).

d. Perform post-hoc test to find which month is different from which month

```r
# Perform Tukey's HSD post-hoc test
tukey_results <- TukeyHSD(anova_results)
print(tukey_results)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = Temp ~ factor(Month), data = airquality)
##
## $`factor(Month)`
##            diff          lwr       upr     p adj
## 6-5 13.55161290   8.84386422 18.259362 0.0000000
## 7-5 18.35483871  13.68583759 23.023840 0.0000000
## 8-5 18.41935484  13.75035372 23.088356 0.0000000
## 9-5 11.35161290   6.64386422 16.059362 0.0000000
## 7-6  4.80322581   0.09547713  9.510974 0.0430674
## 8-6  4.86774194   0.15999325  9.575491 0.0388654
## 9-6 -2.20000000  -6.94617992  2.546180 0.7038121
## 8-7  0.06451613  -4.60448499  4.733517 0.9999995
## 9-7 -7.00322581 -11.71097449 -2.295477 0.0006215
## 9-8 -7.06774194 -11.77549062 -2.359993 0.0005376
```

Interpretation

The Tukey's HSD post-hoc test results indicate significant differences in mean temperature between several pairs of months. We can summarize the findings as follows: - May has the coolest temps compared to all other months — all comparisons with May show significant positive differences.

- June, July, and August are warmer months, with July and August temperatures being very similar (not significantly different).
- September temps drop compared to July and August (significant negative differences).
- September and June temps are not significantly different.

Q.N 8

Do the following: a. Create a dataset with 200 random cases, 1 random binary (1 and 0), variable four randon (cateogorical and contnuous) variable with seed of 39

```r
set.seed(39)
data_8 <- data.frame(
  binary_var = sample(0:1, 200, replace = TRUE),
  categorical_var1 = sample(c("A", "B", "C"), 200, replace = TRUE),
  categorical_var2 = sample(c("X", "Y"), 200, replace = TRUE),
  continuous_var1 = rnorm(200, mean = 50, sd = 10),
```

```
  continuous_var2 = rnorm(200, mean = 100, sd = 20)
)
print(head(data_8))
```

```
##   binary_var categorical_var1 categorical_var2 continuous_var1 continuous_var2
## 1          0                A                X        37.35157        77.84065
## 2          0                C                X        66.21572       108.19520
## 3          1                A                X        34.17535       112.77027
## 4          1                B                X        31.04624        98.37346
## 5          1                C                X        65.58661       122.02813
## 6          0                A                Y        58.72740        95.62703
```

b. Divide into train and test with 70:30 random split

```
library(caret)
```

```
## Loading required package: lattice
```

```
set.seed(39)
# Create a random split of the data into training and testing sets
train_index <- createDataPartition(data_8$binary_var, p = 0.7, list = FALSE)
train_data <- data_8[train_index, ]
test_data <- data_8[-train_index, ]
print(dim(train_data))
```

```
## [1] 140   5
```

```
print(dim(test_data))
```

```
## [1] 60  5
```

c. Fit supervised logistic regression abd decission tree classification model on train data with binary variable as dependent variable and all other variables as independent variables

```
library(rpart)
# Fit logistic regression model
logistic_model <- glm(binary_var ~ ., data = train_data, family = binomial)
print(summary(logistic_model))
```

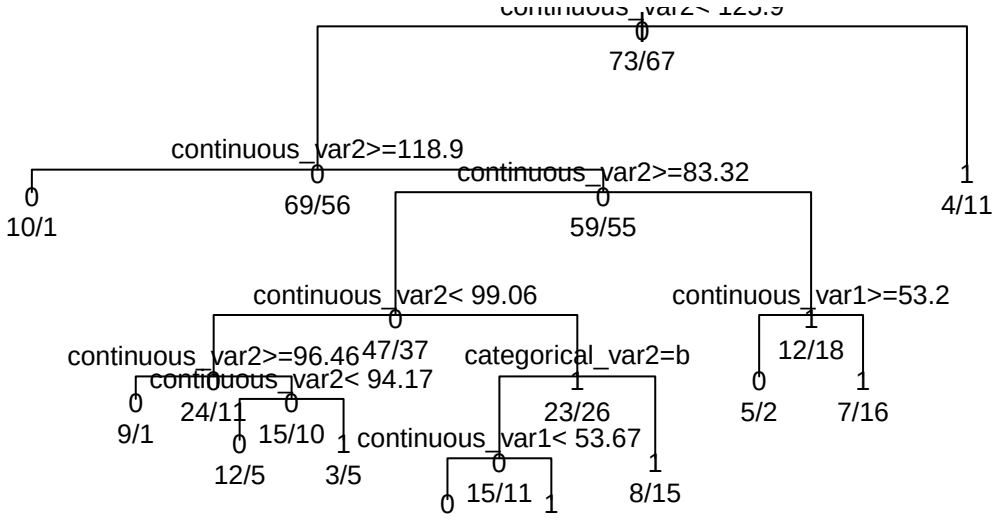
```
##
## Call:
## glm(formula = binary_var ~ ., family = binomial, data = train_data)
##
```

```
## Coefficients:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         0.725287   1.310124   0.554    0.580
## categorical_var1B   0.265633   0.432045   0.615    0.539
## categorical_var1C   0.446847   0.436165   1.024    0.306
## categorical_var2Y  -0.476614   0.348072  -1.369    0.171
## continuous_var1    -0.014794   0.019091  -0.775    0.438
## continuous_var2    -0.001091   0.008901  -0.123    0.902
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 193.82  on 139  degrees of freedom
## Residual deviance: 189.78  on 134  degrees of freedom
## AIC: 201.78
##
## Number of Fisher Scoring iterations: 4
```

```
# Fit decision tree model
decision_tree_model <- rpart(binary_var ~ ., data = train_data, method = "class")
print(decision_tree_model)
```

```
## n= 140
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##  1) root 140 67 0 (0.52142857 0.47857143)
##    2) continuous_var2< 125.9226 125 56 0 (0.55200000 0.44800000)
##      4) continuous_var2>=118.9177 11  1 0 (0.90909091 0.09090909) *
##      5) continuous_var2< 118.9177 114 55 0 (0.51754386 0.48245614)
##       10) continuous_var2>=83.32128 84 37 0 (0.55952381 0.44047619)
##         20) continuous_var2< 99.0636 35 11 0 (0.68571429 0.31428571)
##           40) continuous_var2>=96.45659 10  1 0 (0.90000000 0.10000000) *
##           41) continuous_var2< 96.45659 25 10 0 (0.60000000 0.40000000)
##             82) continuous_var2< 94.17355 17  5 0 (0.70588235 0.29411765) *
##             83) continuous_var2>=94.17355 8  3 1 (0.37500000 0.62500000) *
##         21) continuous_var2>=99.0636 49 23 1 (0.46938776 0.53061224)
##           42) categorical_var2=Y 26 11 0 (0.57692308 0.42307692)
##             84) continuous_var1< 53.67418 18  6 0 (0.66666667 0.33333333) *
##             85) continuous_var1>=53.67418 8  3 1 (0.37500000 0.62500000) *
##           43) categorical_var2=X 23  8 1 (0.34782609 0.65217391) *
##       11) continuous_var2< 83.32128 30 12 1 (0.40000000 0.60000000)
##         22) continuous_var1>=53.19767 7  2 0 (0.71428571 0.28571429) *
##         23) continuous_var1< 53.19767 23  7 1 (0.30434783 0.69565217) *
##    3) continuous_var2>=125.9226 15  4 1 (0.26666667 0.73333333) *
```

```
# Plot the decision tree
plot(decision_tree_model)
text(decision_tree_model, use.n = TRUE, all = TRUE, cex = 0.8)
```



d. Predict the released variable in the test datasets using both model and interpret the results

```
# Predict using logistic regression model
logistic_predictions <- predict(logistic_model, newdata = test_data, type = "response")
# Convert probabilities to binary predictions
logistic_predictions_binary <- ifelse(logistic_predictions > 0.5, 1, 0)
# Predict using decision tree model
decision_tree_predictions <- predict(decision_tree_model, newdata = test_data, type = "class")
# Create a confusion matrix for logistic regression predictions
logistic_confusion_matrix <- table(test_data$binary_var, logistic_predictions_binary)
print(logistic_confusion_matrix)
```

```
##    logistic_predictions_binary
##      0  1
##   0 20 12
##   1 15 13
```

```
# use caret package to calculate accuracy
library(caret)
logistic_accuracy_caret <- confusionMatrix(as.factor(logistic_predictions_binary), as.factor(test_data$
print(logistic_accuracy_caret)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
```

```
##          0 20 15
##          1 12 13
##
##                Accuracy : 0.55
##                  95% CI : (0.4161, 0.6788)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 0.4497
##
##                   Kappa : 0.0899
##
##  Mcnemar's Test P-Value : 0.7003
##
##             Sensitivity : 0.6250
##             Specificity : 0.4643
##          Pos Pred Value : 0.5714
##          Neg Pred Value : 0.5200
##              Prevalence : 0.5333
##          Detection Rate : 0.3333
##    Detection Prevalence : 0.5833
##       Balanced Accuracy : 0.5446
##
##        'Positive' Class : 0
##
```

```
decision_tree_accuracy_caret <- confusionMatrix(as.factor(decision_tree_predictions), as.factor(test_da
print(decision_tree_accuracy_caret)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 13 12
##          1 19 16
##
##                Accuracy : 0.4833
##                  95% CI : (0.3523, 0.6161)
##     No Information Rate : 0.5333
##     P-Value [Acc > NIR] : 0.8175
##
##                   Kappa : -0.022
##
##  Mcnemar's Test P-Value : 0.2812
##
##             Sensitivity : 0.4062
##             Specificity : 0.5714
##          Pos Pred Value : 0.5200
##          Neg Pred Value : 0.4571
##              Prevalence : 0.5333
##          Detection Rate : 0.2167
##    Detection Prevalence : 0.4167
##       Balanced Accuracy : 0.4888
##
##        'Positive' Class : 0
##
```

Interpretation

The logistic regression model achieved an `accuracy of 0.55 ie. 55% with 95% CI : (0.4161, 0.6788)`, while the decision tree model achieved an `accuracy of 0.48 i.e 48% with 95% CI : (0.3523, 0.6161)` on the test dataset.

The specificity and sensitivity for the logistic regression model were 0.46 and 0.62, respectively, while for the decision tree model, they were 0.57 and 0.40, respectively.

Thus oberving the accuracy and other metrics, we can conclude that the logistic regression model performed better than the decision tree model in this case.

Q.N 9

Do the following using inbuilt USArrests dataset: # a. Create a criminality scale of four variables using PCA

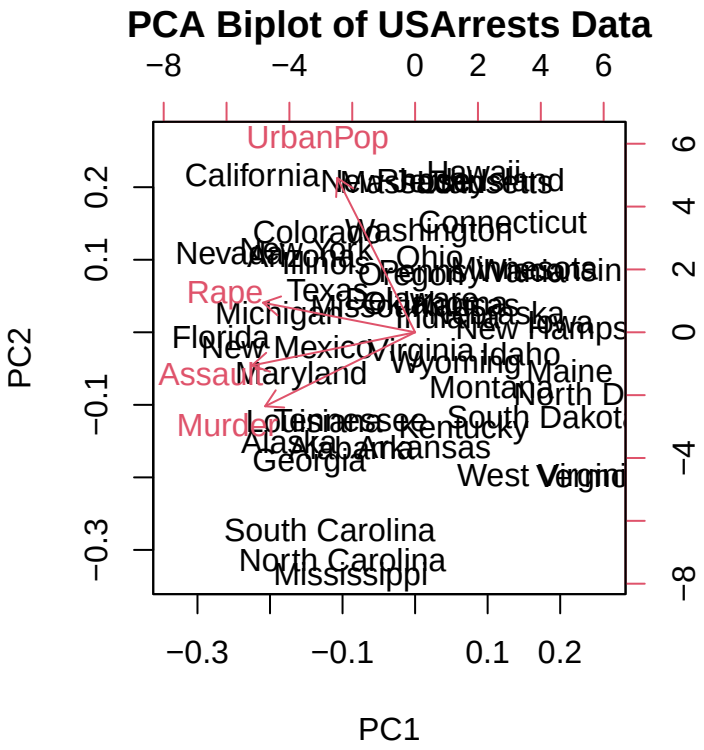
```r
data("USArrests")
print(head(USArrests))
```

```
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

```r
# Scale the data
scaled_data <- scale(USArrests)
# Perform PCA
pca_result <- prcomp(scaled_data, center = TRUE, scale. = TRUE)
print(summary(pca_result))
```

```
## Importance of components:
##                           PC1    PC2     PC3     PC4
## Standard deviation     1.5749 0.9949 0.59713 0.41645
## Proportion of Variance 0.6201 0.2474 0.08914 0.04336
## Cumulative Proportion  0.6201 0.8675 0.95664 1.00000
```

```r
# biplot of PCA
biplot(pca_result, main = "PCA Biplot of USArrests Data")
```



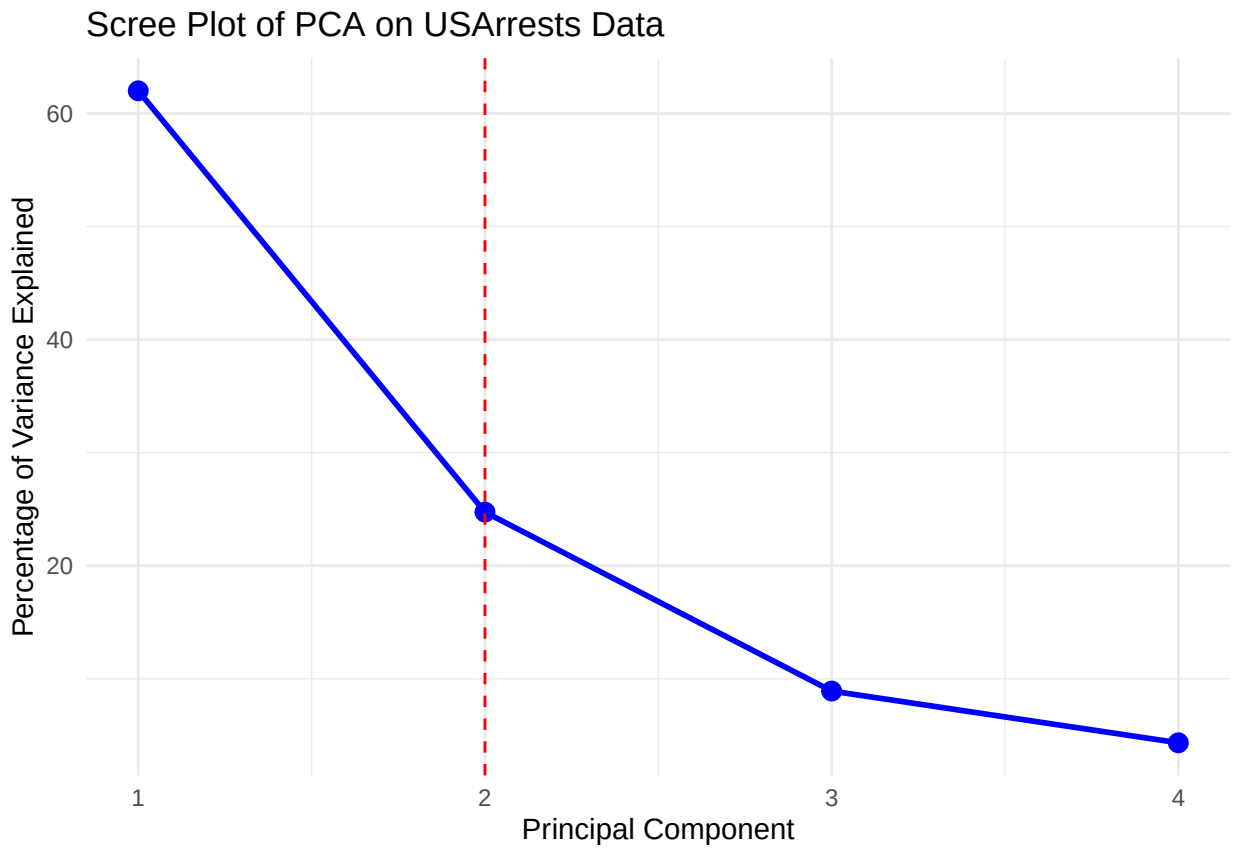
Interpretation , from the result we can see PC1 has higher Standard deviation(SD) square of which give eigen value > 1 . and the variance explainded is 62%. So, we can take it but needs confirmation with other test.

b. Check the eigen value and interpret using kaiser criterion

“‘{r question9b}s eigen_values <- pca_result$sdev^2 print(eigen_values)

```
## Interpretation
The first component has eigen value (2.48) which is greater than 1, indicating that it explains more va

# c. Check the scree plot and interpret

‘‘‘ r
library(ggplot2)
eigen_values <- pca_result$sdev^2
explained_var <- eigen_values / sum(eigen_values) * 100
components <- 1:length(eigen_values)

scree_data <- data.frame(
  PC = components,
  Variance = explained_var
)

ggplot(scree_data, aes(x = PC, y = Variance)) +
```

```
  geom_point(color = "blue", size = 3) +
  geom_line(color = "blue", linewidth = 1) +
  geom_vline(xintercept = which.max(diff(diff(explained_var)) < 0)[1] + 1,
             linetype = "dashed", color = "red") +
  labs(
    title = "Scree Plot of PCA on USArrests Data",
    x = "Principal Component",
    y = "Percentage of Variance Explained"
  ) +
  theme_minimal()
```



Interpretation

From the scree plot we can take 2 components as there is sharp bend at 2.

#d. Revise the criminality scale usign VARIMAX rotation and interpret

```
library(psych)
```

```
##
## Attaching package: 'psych'

## The following object is masked from 'package:car':
##
##     logit
```

```
## The following objects are masked from 'package:ggplot2':
##
##     %+%, alpha
```

```r
# Perform PCA with VARIMAX rotation
pca_varimax <- principal(scaled_data, nfactors = 2, rotate = "varimax")
print(pca_varimax)
```

```
## Principal Components Analysis
## Call: principal(r = scaled_data, nfactors = 2, rotate = "varimax")
## Standardized loadings (pattern matrix) based upon correlation matrix
##           RC1   RC2   h2    u2 com
## Murder   0.94 -0.06 0.89 0.115 1.0
## Assault  0.92  0.18 0.88 0.121 1.1
## UrbanPop 0.07  0.97 0.95 0.054 1.0
## Rape     0.73  0.48 0.76 0.240 1.7
##
##                        RC1  RC2
## SS loadings           2.26 1.21
## Proportion Var        0.57 0.30
## Cumulative Var        0.57 0.87
## Proportion Explained  0.65 0.35
## Cumulative Proportion 0.65 1.00
##
## Mean item complexity =  1.2
## Test of the hypothesis that 2 components are sufficient.
##
## The root mean square of the residuals (RMSR) is  0.08
##  with the empirical chi square  3.44  with prob <  NA
##
## Fit based upon off diagonal values = 0.98
```

Interpretation

Factor 1 loads highly on Murder and Assault, represents Violent Crime.

Factor 2 loads on UrbanPop and Rape, represents Urban-Related Crime or Sexual/Population-linked Crime.

The VARIMAX rotation helped in clearly separating Murder/Assault from UrbanPop/Rape, making the latent criminality scales more interpretable.

Q.N 10

a. Create a data set with 200 random cases and 5 random variabless with 39 as seed

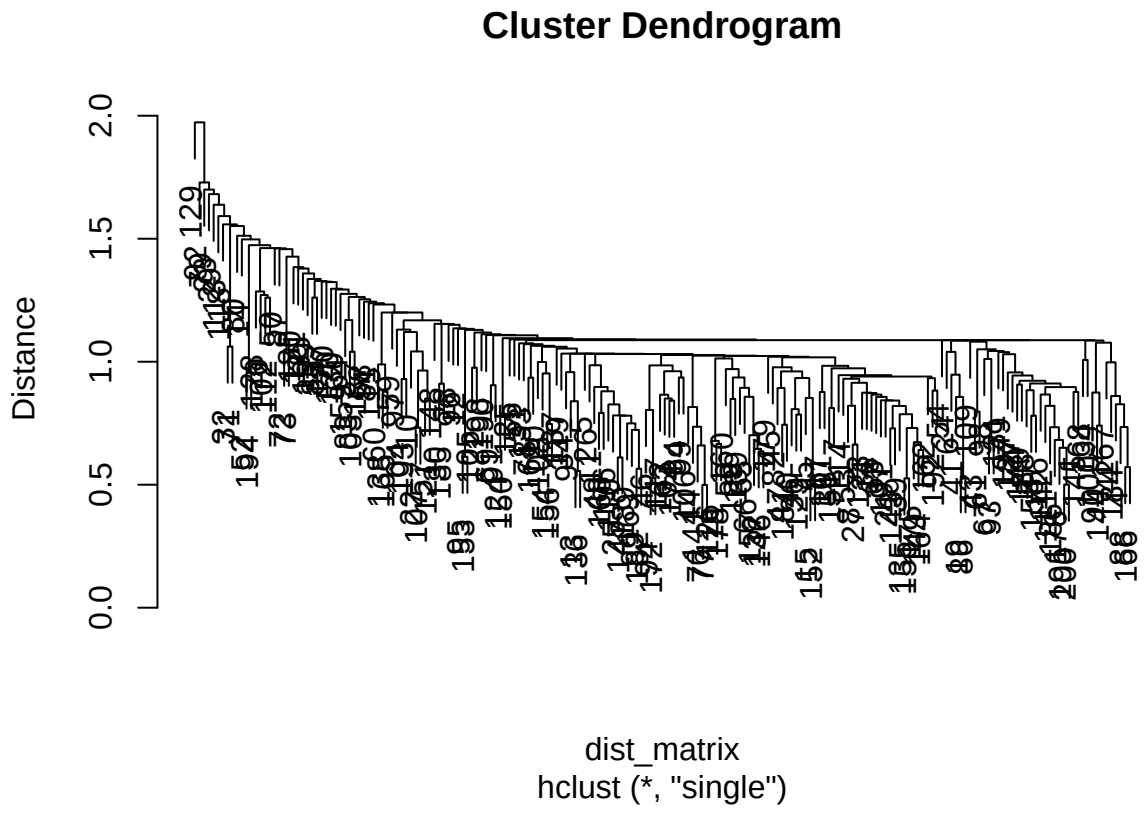
```r
set.seed(39)

# Generate dataset: 200 rows and 5 columns of random numbers
random_data <- as.data.frame(matrix(rnorm(200 * 5), nrow = 200, ncol = 5))

# Optionally, name the variables
colnames(random_data) <- paste0("Var", 1:5)
```

```r
# View the first few rows
head(random_data)
```

```
##         Var1       Var2       Var3        Var4      Var5
## 1 -0.1855657  0.3671881  1.6871100  0.48455419 0.9180654
## 2 -1.2292427 -0.7341742  0.4130404 -0.48351527 1.3869316
## 3 -0.4272029  0.2993456 -0.2695057  1.00847144 0.7511314
## 4 -0.5959819 -0.3400983  1.8062308 -0.02909403 0.1268994
## 5  0.4673236 -0.3427875 -0.4272769 -0.94365395 2.4080354
## 6  0.4216390  1.3538053  0.4087177  1.81956822 0.5780611
```

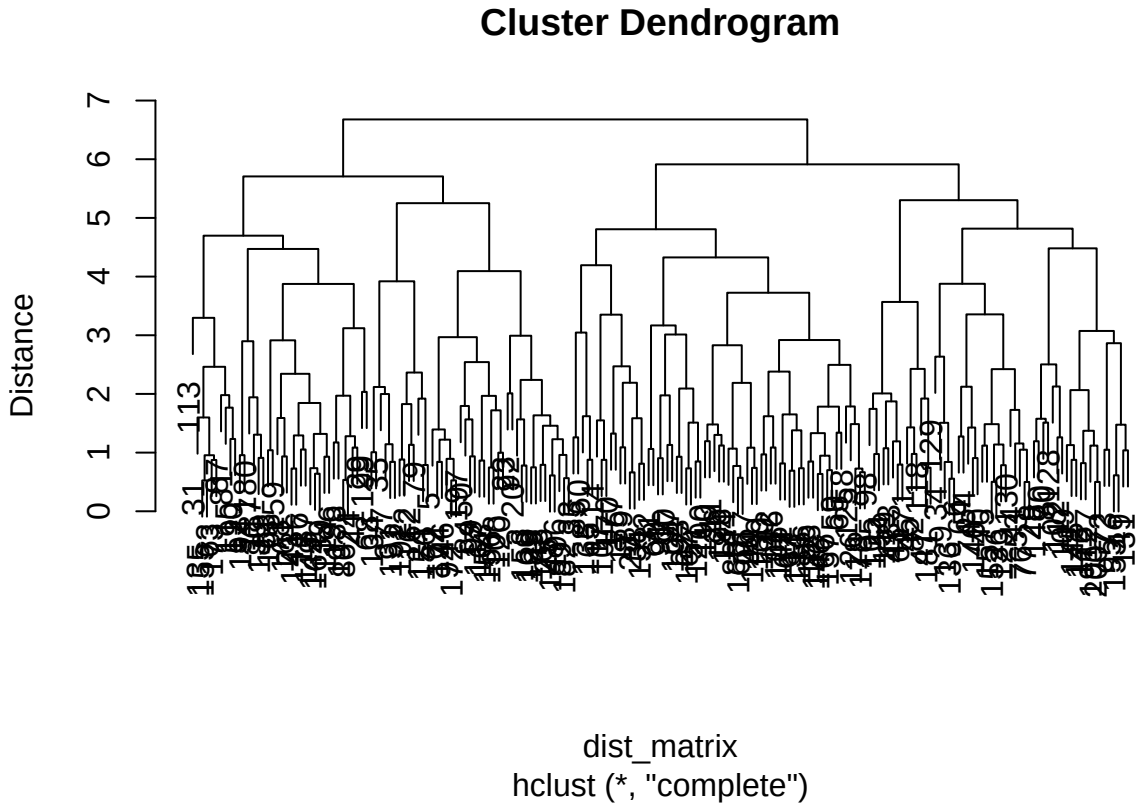
b. Fit hca with single linkage

```r
dist_matrix <- dist(random_data)

# Single linkage
hc_single <- hclust(dist_matrix, method = "single")
# plot
plot(
  hc_single,
  labels=rownames(random_data),
  ylab="Distance"
)
```



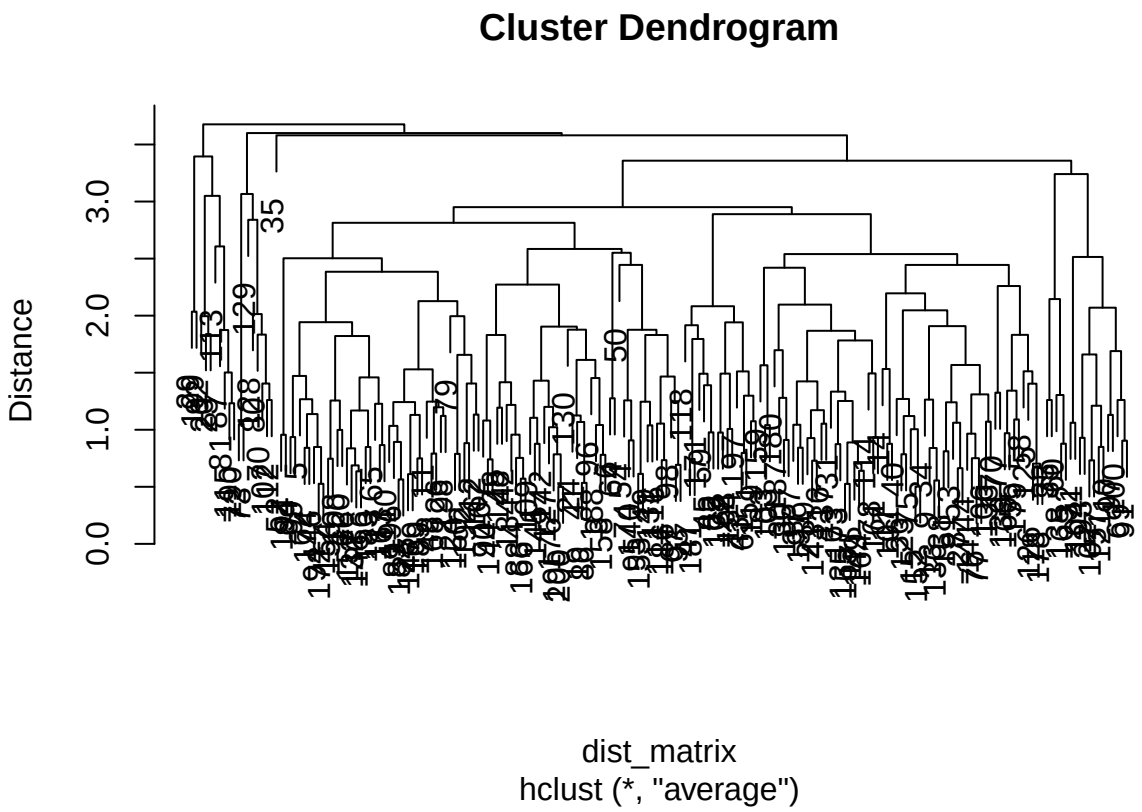
c. Fit hca with complete

```
# Complete linkage
hc_complete <- hclust(dist_matrix, method = "complete")

plot(
  hc_complete,
  labels=rownames(random_data),
  ylab="Distance"
)
```



d. Fit hca with average

```
# Average linkage
hc_average <- hclust(dist_matrix, method = "average")

plot(
  hc_average,
  labels=rownames(random_data),
  ylab="Distance"
)
```



d. Find best method

```
# Compute cophenetic distance matrices
cor_single <- cor(cophenetic(hc_single), dist_matrix)
cor_complete <- cor(cophenetic(hc_complete), dist_matrix)
cor_average <- cor(cophenetic(hc_average), dist_matrix)

# Display correlation coefficients
print(c(Single = cor_single, Complete = cor_complete, Average = cor_average))
```

```
##    Single  Complete   Average
## 0.4920251 0.3793040 0.5521490
```

Interpretation

So we can see from the dendogram the best number of clusters can be 4, as the distance between the clusters is highest at that point as seen in the plot for complete.