1. **Write a R program to create an array of two 3x3 matrices each with 3 rows and 3 columns from two given two vectors. Print the second row of the second matrix of the array and the element in the 3rd row and 3rd column of the 1st matrix.**

   R-CODE:

```r
v1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
v2 <- c(10, 11, 12, 13, 14, 15, 16, 17, 18)
arr <- array(c(v1, v2), dim = c(3, 3, 2))
print(arr[2, , 2])
print(arr[3, 3, 1])
```

   OUTPUT:

```
Console   Terminal   Background Jobs
R 4.4.2 · ~/
> v1 <- c(1, 2, 3, 4, 5, 6, 7, 8, 9)
> v2 <- c(10, 11, 12, 13, 14, 15, 16, 17, 18)
> arr <- array(c(v1, v2), dim = c(3, 3, 2))
> print(arr[2, , 2])
[1] 11 14 17
> print(arr[3, 3, 1])
[1] 9
>
>
```

2. **Write a R program to create an array using four given columns, three given rows, and two given tables and display the content of the array.**

   R-CODE:

```r
v <- c(1:24)
arr <- array(v, dim = c(3, 4, 2))
print(arr)
```

OUTPUT:

```
R 4.4.2 · ~/
> v <- c(1:24)
> arr <- array(v, dim = c(3, 4, 2))
> print(arr)
, , 1

     [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12

, , 2

     [,1] [,2] [,3] [,4]
[1,]   13   16   19   22
[2,]   14   17   20   23
[3,]   15   18   21   24

> |
```

3. **Write a R program to create a factor corresponding to height of women data set , which inbuild in R, contains height and weights for a sample of women.**

R-CODE:

```
data(women)
height_factor <- factor(women$height)
print(height_factor)
```

OUTPUT:

```
Console   Terminal ×   Background Jobs ×
R 4.4.2 · ~/
> data(women)
> height_factor <- factor(women$height)
> print(height_factor)
 [1] 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
Levels: 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
>
> |
```

4. **Write a R program to extract the five of the levels of factor created from a random sample from the LETTERS (Part of the base R distribution.)**

R-CODE:

```r
set.seed(123)
sample_letters <- sample(LETTERS, 10, replace = TRUE)
factor_letters <- factor(sample_letters)
print(levels(factor_letters)[1:5])
```

OUTPUT:

```
R 4.4.2 · ~/
> set.seed(123)
> sample_letters <- sample(LETTERS, 10, replace = TRUE)
> factor_letters <- factor(sample_letters)
> print(levels(factor_letters)[1:5])
[1] "C" "E" "J" "K" "N"
>
>
```

5. **Iris dataset is a very famous dataset in almost all data mining, machine learning courses, and it has been an R build-in dataset. The dataset consists of 50 samples from each of three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor). Four features(variables) were measured from each sample, they are the length and the width of sepal and petal, in centimetres. Perform the following EDA steps . (i)Find dimension, Structure, Summary statistics, Standard Deviation of all features. (ii)Find mean and standard deviation of features groped by three species of Iris flowers (Iris setosa, Iris virginica and Iris versicolor) (iii)Find quantile value of sepal width and length (iV)create new data frame named iris1 which have a new column name Sepal.Length.Cate that categorizes “Sepal.Length” by quantile (V) Average value of numerical varialbes by two categorical variables: Species and Sepal.Length.Cate: (vi) Average mean value of**

numerical varialbes by Species and Sepal.Length.Cate (vii)Create Pivot Table based on Species and Sepal.Length.Cate.

R-CODE:

```r
aggregate(iris1[, 1:4], by = list(Species = iris1$Species, Sepal.Length.Cate = iris1$Sepal.Length.Cate), FUN = mean)
```

OUTPUT:

```
>
> aggregate(iris1[, 1:4], by = list(Species = iris1$Species, Sepal.Length.Cate = iris1$Sepal.Length.Cate), FUN = mean)
      Species Sepal.Length.Cate Sepal.Length Sepal.Width Petal.Length Petal.Width
1      setosa         [4.3,5.1]     4.838889    3.291667     1.455556   0.2416667
2  versicolor         [4.3,5.1]     5.000000    2.300000     3.275000   1.0250000
3   virginica         [4.3,5.1]     4.900000    2.500000     4.500000   1.7000000
4      setosa         (5.1,5.8]     5.435714    3.778571     1.478571   0.2571429
5  versicolor         (5.1,5.8]     5.600000    2.705000     4.055000   1.2400000
6   virginica         (5.1,5.8]     5.740000    2.700000     5.040000   2.0400000
7  versicolor         (5.8,6.4]     6.135294    2.835294     4.511765   1.4294118
8   virginica         (5.8,6.4]     6.238889    2.900000     5.283333   1.9222222
9  versicolor         (6.4,7.9]     6.722222    3.000000     4.677778   1.4555556
10  virginica         (6.4,7.9]     7.057692    3.096154     5.876923   2.1076923
>
```

6. **Randomly Sample the iris dataset such as 80% data for training and 20% for test and create Logistics regression with train data, use species as target and petals width and length as feature variables , Predict the probability of the model using test data, Create Confusion matrix for above test model**

   R-CODE:

Source on Save

```r
library(nnet)

set.seed(123)
index <- sample(1:nrow(iris), 0.8 * nrow(iris))
train_data <- iris[index, ]
test_data <- iris[-index, ]

model <- multinom(Species ~ Petal.Width + Petal.Length, data = train_data)

pred_prob <- predict(model, test_data, type = "prob")
pred_class <- predict(model, test_data)

table(Predicted = pred_class, Actual = test_data$Species)
```

OUTPUT:

R 4.4.2 · ~/

```
> library(nnet)
> set.seed(123)
> index <- sample(1:nrow(iris), 0.8 * nrow(iris))
> train_data <- iris[index, ]
> test_data <- iris[-index, ]
> model <- multinom(Species ~ Petal.Width + Petal.Length, data = train_data)
# weights:  12 (6 variable)
initial  value 131.833475
iter  10 value 10.207694
iter  20 value 8.641476
iter  30 value 8.623607
iter  40 value 8.609503
iter  50 value 8.602741
iter  60 value 8.600893
iter  70 value 8.596472
iter  80 value 8.594230
iter  90 value 8.593757
iter 100 value 8.592407
final  value 8.592407
stopped after 100 iterations
> pred_prob <- predict(model, test_data, type = "prob")
> pred_class <- predict(model, test_data)
> table(Predicted = pred_class, Actual = test_data$Species)
            Actual
Predicted    setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         14         0
  virginica       0          1         5
> 
```

7. **Explore the airquality dataset. It contains daily air quality measurements from New York during a period of five months: • Ozone: mean ozone concentration (ppb), • Solar.R: solar radiation (Langley), • Wind: average wind speed (mph), • Temp: maximum daily temperature in degrees Fahrenheit, • Month: numeric month (May=5, June=6, and so on),• Day: numeric day of the month (1 31). i. Compute the mean temperature(don't use build in function) ii.Extract the first five rows from airquality. iii.Extract all columns from airquality except Temp and Wind iv.Which was the coldest day during the period? v.How many days was the wind speed greater than 17 mph?**

R-CODE:

```r
data(airquality)

mean_temp <- sum(airquality$Temp, na.rm = TRUE) / sum(!is.na(airquality$Temp))
print(mean_temp)

head(airquality, 5)

airquality_subset <- airquality[, !names(airquality) %in% c("Temp", "Wind")]
print(airquality_subset)

coldest_day <- airquality[which.min(airquality$Temp), ]
print(coldest_day)

wind_count <- sum(airquality$Wind > 17, na.rm = TRUE)
print(wind_count)
```

OUTPUT:

R 4.4.2 · ~/

```
121   118      225      8  29
122    84      237      8  30
123    85      188      8  31
124    96      167      9   1
125    78      197      9   2
126    73      183      9   3
127    91      189      9   4
128    47       95      9   5
129    32       92      9   6
130    20      252      9   7
131    23      220      9   8
132    21      230      9   9
133    24      259      9  10
134    44      236      9  11
135    21      259      9  12
136    28      238      9  13
137     9       24      9  14
138    13      112      9  15
139    46      237      9  16
140    18      224      9  17
141    13       27      9  18
142    24      238      9  19
143    16      201      9  20
144    13      238      9  21
145    23       14      9  22
146    36      139      9  23
147     7       49      9  24
148    14       20      9  25
149    30      193      9  26
150    NA      145      9  27
151    14      191      9  28
152    18      131      9  29
153    20      223      9  30
> coldest_day <- airquality[which.min(airquality$Temp), ]
> print(coldest_day)
  Ozone Solar.R Wind Temp Month Day
5    NA      NA 14.3   56     5   5
> wind_count <- sum(airquality$Wind > 17, na.rm = TRUE)
> print(wind_count)
[1] 3
>
```

8. (i)Get the Summary Statistics of air quality dataset (ii)Melt airquality data set and display as a long – format data? (iii)Melt airquality data and specify month and day to be "ID variables"? (iv)Cast the molten airquality data set with respect to month and date features (v) Use cast function appropriately and compute the average of Ozone, Solar.R , Wind and temperature per month?

R-CODE:

ERS.R × | MULTIPLE LINES IN LINE CHART.R × | LINEAR REGRESSION.R × | BOXPLOT FOR MPG AND CYLINDER.R × | BOX PLOT TENNIS SCORES.R × | Untitled

Source on Save | Run

```r
data(airquality)

summary(airquality)

library(reshape2)

molten_data <- melt(airquality)
print(molten_data)

molten_data_id <- melt(airquality, id.vars = c("Month", "Day"))
print(molten_data_id)

cast_data <- dcast(molten_data_id, Month + Day ~ variable)
print(cast_data)

avg_per_month <- dcast(molten_data_id, Month ~ variable, fun.aggregate = mean, na.rm = TRUE)
print(avg_per_month)
```

18:1 (Top Level) ÷

OUTPUT:

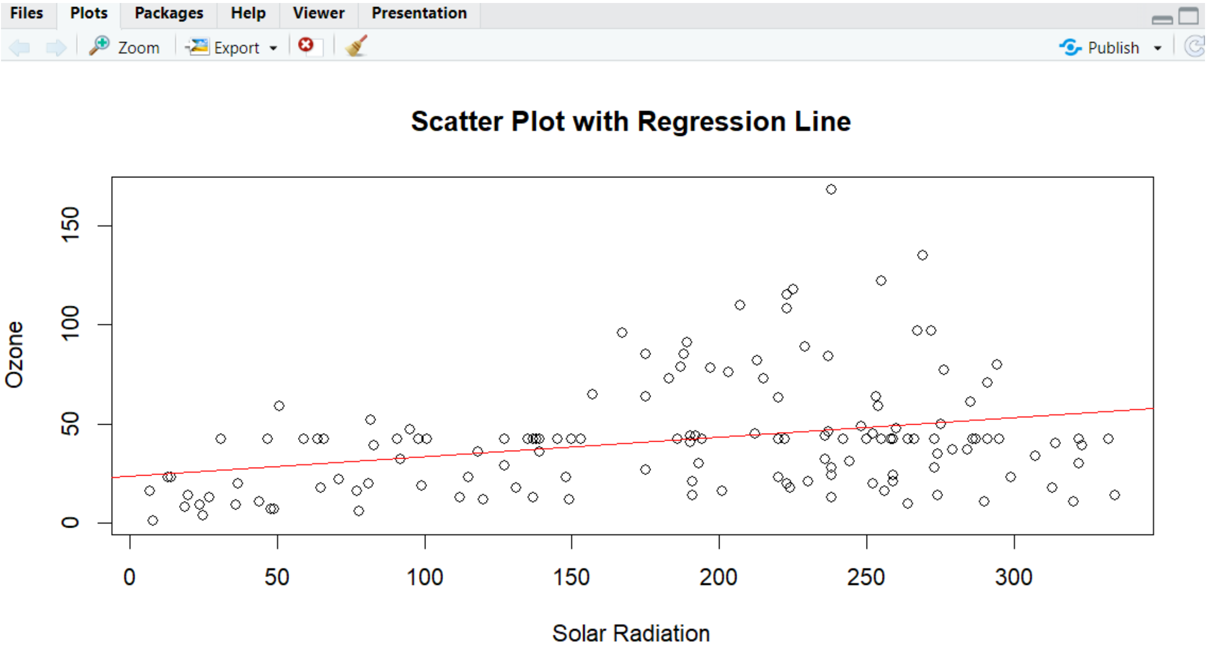
```
avg_per_month <- dcast(molten_data_id, Month ~ variable, fun.aggregate = mean, na.rm = TRUE)
print(avg_per_month)
Month    Ozone  Solar.R      Wind     Temp
    5 23.61538 181.2963 11.622581 65.54839
    6 29.44444 190.1667 10.266667 79.10000
    7 59.11538 216.4839  8.941935 83.90323
    8 59.96154 171.8571  8.793548 83.96774
    9 31.44828 167.4333 10.180000 76.90000
```

9. .(i) Find any missing values(na) in features and drop the missing values if its less than 10% else replace that with mean of that feature. (ii) Apply a linear regression algorithm using Least Squares Method on "Ozone" and "Solar.R" (iii)Plot Scatter plot between Ozone and Solar and add regression line created by above model.

R-CODE:

```r
data(airquality)

na_counts <- colSums(is.na(airquality))
total_rows <- nrow(airquality)

for (col in names(airquality)) {
  if (na_counts[col] > 0) {
    if ((na_counts[col] / total_rows) < 0.1) {
      airquality <- airquality[!is.na(airquality[[col]]), ]
    } else {
      airquality[[col]][is.na(airquality[[col]])] <- mean(airquality[[col]], na.rm = TRUE)
    }
  }
}

model <- lm(Ozone ~ Solar.R, data = airquality)

plot(airquality$Solar.R, airquality$Ozone, xlab = "Solar Radiation", ylab = "Ozone", main = "Scatter Plot with Regression Line")
abline(model, col = "red")
```

OUTPUT:



10. Explore the USArrests dataset, contains the number of arrests for murder, assault, and rape for each of the 50 states in 1973. It also contains the percentage of people in the state who live in an urban area. (i) a. Explore the summary of Data set, like number of Features and its type. Find the number of records for each feature. Print the statistical feature of data b. Print the state which saw the largest total number of rape c. Print the states with the max & min crime rates for murder (ii).a. Find the correlation among the features b. Print the states which have assault arrests more than median of the country c. Print the states are in the bottom 25% of murder (iii). a. Create a histogram and density plot of murder arrests by US stat b. Create the plot that shows the relationship between murder arrest rate and proportion of the population that is urbanised by state. Then enrich the

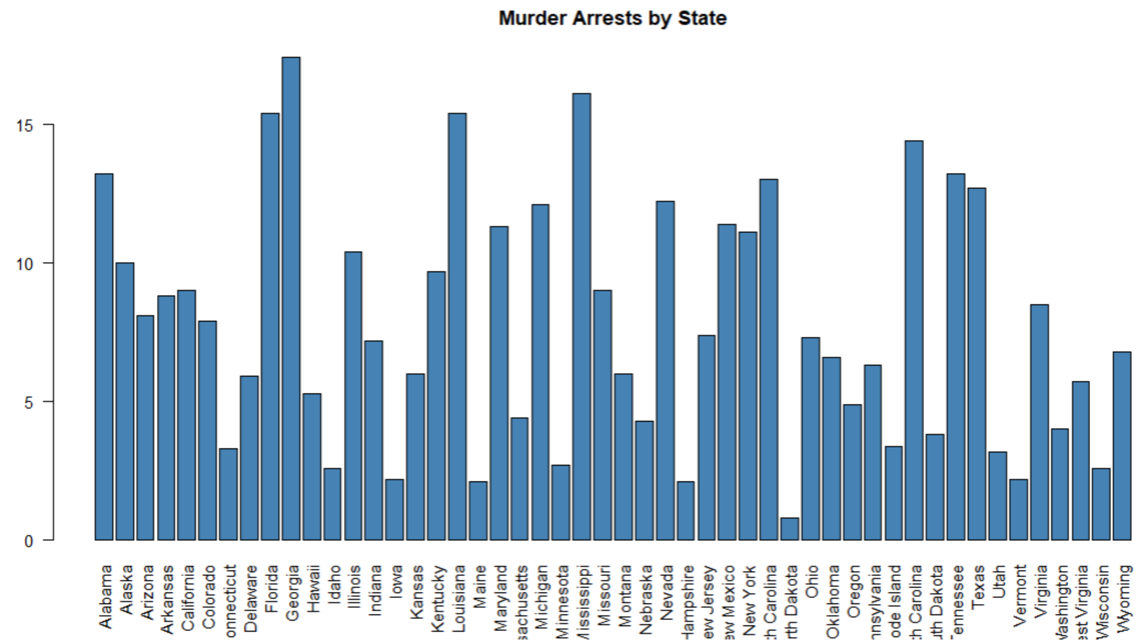
chart by adding assault arrest rates (by colouring the points from blue (low) to red (high)). c. Draw a bar graph to

R-CODE:

```r
data(USArrests)

summary(USArrests)
str(USArrests)
sapply(USArrests, length)

state_largest_rape <- rownames(USArrests)[which.max(USArrests$Rape)]
print(state_largest_rape)

state_max_murder <- rownames(USArrests)[which.max(USArrests$Murder)]
state_min_murder <- rownames(USArrests)[which.min(USArrests$Murder)]
print(state_max_murder)
print(state_min_murder)

correlation_matrix <- cor(USArrests)
print(correlation_matrix)

assault_median <- median(USArrests$Assault)
high_assault_states <- rownames(USArrests)[USArrests$Assault > assault_median]
print(high_assault_states)

bottom_25_murder <- quantile(USArrests$Murder, 0.25)
low_murder_states <- rownames(USArrests)[USArrests$Murder <= bottom_25_murder]
print(low_murder_states)

hist(USArrests$Murder, main = "Histogram of Murder Arrests", xlab = "Murder Arrests", col = "blue", border = "black")
plot(density(USArrests$Murder), main = "Density Plot of Murder Arrests")

library(ggplot2)
ggplot(USArrests, aes(x = UrbanPop, y = Murder, color = Assault)) +
  geom_point() +
  scale_color_gradient(low = "blue", high = "red") +
  labs(title = "Murder Arrests vs Urban Population", x = "Urban Population", y = "Murder Arrests")

barplot(USArrests$Murder, names.arg = rownames(USArrests), las = 2, col = "steelblue", main = "Murder Arrests by State")
```

OUTPUT:

```
> data(USArrests)
> summary(USArrests)
     Murder          Assault         UrbanPop          Rape      
 Min.   : 0.800   Min.   : 45.0   Min.   :32.00   Min.   : 7.30  
 1st Qu.: 4.075   1st Qu.:109.0   1st Qu.:54.50   1st Qu.:15.07  
 Median : 7.250   Median :159.0   Median :66.00   Median :20.10  
 Mean   : 7.788   Mean   :170.8   Mean   :65.54   Mean   :21.23  
 3rd Qu.:11.250   3rd Qu.:249.0   3rd Qu.:77.75   3rd Qu.:26.18  
 Max.   :17.400   Max.   :337.0   Max.   :91.00   Max.   :46.00  
> str(USArrests)
'data.frame':	50 obs. of  4 variables:
 $ Murder  : num  13.2 10 8.1 8.8 9 7.9 3.3 5.9 15.4 17.4 ...
 $ Assault : int  236 263 294 190 276 204 110 238 335 211 ...
 $ UrbanPop: int  58 48 80 50 91 78 77 72 80 60 ...
 $ Rape    : num  21.2 44.5 31 19.5 40.6 38.7 11.1 15.8 31.9 25.8 ...
> sapply(USArrests, length)
  Murder  Assault UrbanPop     Rape 
      50       50       50       50 
> state_largest_rape <- rownames(USArrests)[which.max(USArrests$Rape)]
> print(state_largest_rape)
[1] "Nevada"
> state_max_murder <- rownames(USArrests)[which.max(USArrests$Murder)]
> state_min_murder <- rownames(USArrests)[which.min(USArrests$Murder)]
> print(state_max_murder)
[1] "Georgia"
> print(state_min_murder)
```


Murder Arrests by State
0
5
10
15
Alabama
Alaska
Arizona
Arkansas
California
Colorado
onnecticut
Delaware
Florida
Georgia
Hawaii
Idaho
Illinois
Indiana
Iowa
Kansas
Kentucky
Louisiana
Maine
Maryland
sachusetts
Michigan
Minnesota
Mississippi
Missouri
Montana
Nebraska
Nevada
Hampshire
ew Jersey
ew Mexico
New York
h Carolina
rth Dakota
Ohio
Oklahoma
Oregon
nnsylvania
ode Island
h Carolina
uth Dakota
Tennessee
Texas
Utah
Vermont
Virginia
Vashington
est Virginia
Wisconsin
Wyoming