UNIT–3 : Essentials of R (PYQ Solution)

PART–A Answers (≈70 Words Each)

Q1. What are the basic data types in R? (Expected)

R has five basic (atomic) data types:

- **numeric: decimal numbers, e.g. 3.14**
- **integer: whole numbers, written with L, e.g. 5L**
- **character: text, e.g. "data"**
- **logical: TRUE or FALSE**
- **complex: e.g. 2+3i**

The type of a value is checked with class() or typeof(). R is case-sensitive and assigns values with <–.

Q2. What is a data frame? How is it created in R? (Expected)

A data frame is a two-dimensional table in which every column is a vector of the same length, and different columns may have different types. It is the standard object for datasets in R.

```
df <- data.frame(id = 1:3, name = c("A", "B", "C"), marks = c(72, 58, 91))
```

Useful functions: str(df), summary(df), head(df), nrow(df).

Q3. What is one-hot encoding? (Expected)

One-hot encoding converts a categorical variable into several binary columns, one for each category, where exactly one column is 1 and the others are 0. For colour = red, green: red becomes (1, 0) and green becomes (0, 1).

It is used for unordered categories so that the model does not assume a false order. In R: model.matrix(~ colour - 1) or the caret function dummyVars().

PART–B Answers (≈250 Words Each)

Q1. Explain the control structures in R with examples. (Expected)

Control structures decide the order in which statements execute.

Structure	Purpose	Example
if / else	Decision	if (m >= 40) print("Pass") else print("Fail")
ifelse()	Vectorised decision	ifelse(x > 0, "pos", "neg")
for	Repeat a fixed number of times	for (i in 1:5) print(i^2)
while	Repeat while a condition is TRUE	while (i <= 3) { print(i); i <- i + 1 }
repeat	Repeat until break	repeat { i <- i + 1; if (i > 5) break }
switch	Choose one of many cases	switch("b", a = "one", b = "two")
break / next	Leave loop / skip one iteration	if (i == 3) next
function	Reusable block	sq <- function(n) n^2

```
total <- 0
for (i in 1:10) {
  if (i %% 2 == 0) next      # skip even numbers
  total <- total + i
}
print(total)                 # 25
```

Vectorised functions (sapply, ifelse) are preferred to loops in R because they are faster and shorter.

Q2. What is feature scaling? Apply min-max scaling and standardisation to a small dataset. (Expected)

Feature scaling brings numeric features to a comparable range so that a feature with large values does not dominate distance-based or gradient-based models such as k-means, kNN, SVM and neural networks. Decision trees do not need it.

$$\text{Min-max: } x' = \frac{x - x_{min}}{x_{max} - x_{min}} \qquad \text{Z-score: } z = \frac{x - \mu}{\sigma}$$

Data: 10, 20, 30, 40, 100

Min-max: $x_{min} = 10$, $x_{max} = 100$, range 90

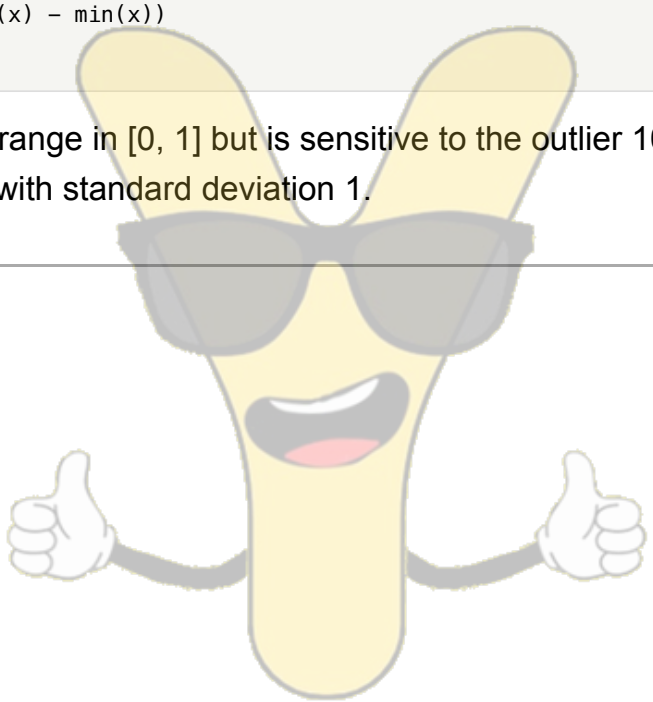
x	10	20	30	40	100
x'	0	0.111	0.222	0.333	1

Standardisation: $\mu = 200/5 = 40$.
Sum of squared deviations $= 900 + 400 + 100 + 0 + 3600 = 5000$; sample variance $= 5000/4 = 1250$; $\sigma = 35.36$.

x	10	20	30	40	100
z	-0.849	-0.566	-0.283	0	1.697

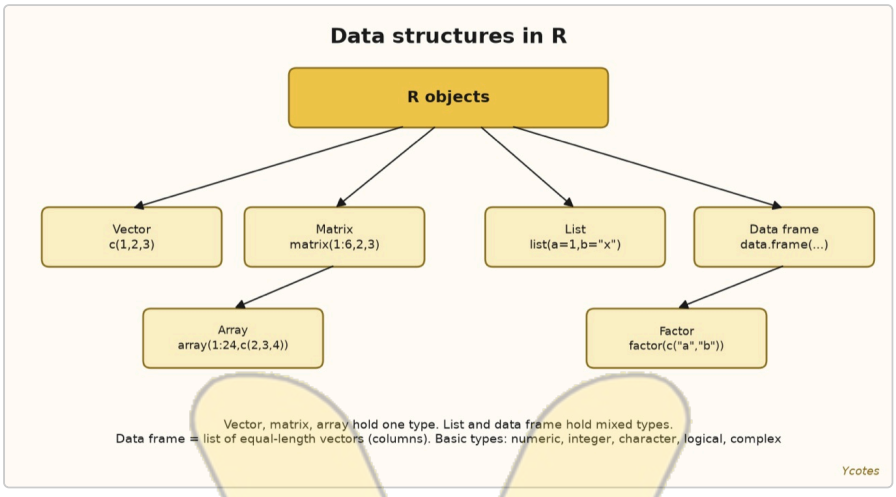
```
x <- c(10, 20, 30, 40, 100)
(x - min(x)) / (max(x) - min(x))
scale(x)
```

Min-max keeps the range in [0, 1] but is sensitive to the outlier 100; z-score centres the data at mean 0 with standard deviation 1.



PART–C Answers (≈500 Words Each)

Q1. Explain the data objects of R (vector, list, matrix, array, factor, data frame) with code. (Expected)



Everything in R is an object. The main data objects are:

1. Vector: one-dimensional, all elements of the same type.

```
v <- c(10, 20, 30);  v[2];  v * 2;  length(v)
```

2. List: an ordered collection that can hold different types, even vectors and other lists.

```
l <- list(name = "Asha", age = 21, marks = c(70, 80))
l[["age"]];  l[["marks"]][2]
```

3. Matrix: two-dimensional, one type.

```
m <- matrix(1:6, nrow = 2, ncol = 3);  m[1, 3];  t(m);  m %*% t(m)
```

4. Array: n-dimensional, one type.

```
a <- array(1:24, dim = c(2, 3, 4));  a[1, 2, 3]
```

5. Factor: categorical data with fixed levels, stored as integers.

```
f <- factor(c("low", "high", "low"), levels = c("low", "high"));  levels(f);  table(f)
```

6. Data frame: table of equal-length columns of mixed types.

```
df <- data.frame(id = 1:3, marks = c(70, 55, 90))
df[["marks"]];  df[2, ];  df[df[["marks"]] > 60, ]
```

Comparison:

Object	Dimensions	Mixed types?	Typical use
Vector	1	No	Single variable
List	1	Yes	Return several results
Matrix	2	No	Numerical computation
Array	n	No	Image or cube data
Factor	1	No (categories)	Categorical variable
Data frame	2	Yes	Datasets

Useful checks: class(), str(), length(), dim(). A data frame is internally a list of vectors of equal length.

Q2. Explain feature engineering: label encoding, one-hot encoding, scaling and dimensionality reduction (PCA) with R code. (Expected)

Feature engineering is the process of creating, transforming and selecting input variables so that a model can learn better. Good features often improve a model more than a better algorithm.

1. Label encoding: replace each category with an integer. Right for **ordinal** data where order has meaning.

```
size <- factor(c("small", "large", "medium"), levels = c("small", "medium", "large"))
as.integer(size)        # 1 3 2
```

2. One-hot encoding: one 0/1 column per category; right for **nominal** data.

```
colour <- factor(c("red", "green", "red"))
model.matrix(~ colour - 1)
```

colour	red	green
red	1	0
green	0	1
Label-encoding a nominal variable would give a false order (green = 2 > red = 1).		

3. Scaling: min-max $(x - x_{min})/(x_{max} - x_{min})$ or standardisation $(x - \mu)/\sigma$.

```
scaled <- scale(df[ , c("age", "income")])
```

4. Dimensionality reduction (PCA): project the data on new uncorrelated axes (principal components) ordered by the variance they keep. Steps: standardise, compute covariance matrix, find eigenvectors and eigenvalues, keep the top components.

```
pc <- prcomp(iris[ , 1:4], scale. = TRUE)
summary(pc)               # PC1 + PC2 explain about 96% of the variance
reduced <- pc[["x"]][ , 1:2]
```

Benefits of reduction: less computation, less overfitting, easier visualisation and removal of correlated features (curse of dimensionality). **Other feature tasks:** creating new columns (BMI from height and weight), binning, log transforms and feature selection.

Conclusion: encode categories correctly, scale numeric values, and reduce dimensions only when needed, always using only the training data to learn the transformation.
