

**Question 1**

R was developed by statisticians working at

- (i) Insightful
- (ii) Bell Labs
- (iii) The University of New South Wales
- (iv) The University of Auckland

## Question 2

The definition of free software consists of four freedoms (freedoms 0 through 3).

Which of the following is NOT one of the freedoms that are part of the definition?

- (i) The freedom to study how the program works, and adapt it to your needs.
- (ii) The freedom to improve the program, and release your improvements to the public, so that the whole community benefits.
- (iii) The freedom to run the program, for any purpose.
- (iv) The freedom to sell the software for any price.

### Question 3

In **R** the following are all **atomic data types** EXCEPT

- (i) `logical`
- (ii) `complex`
- (iii) `list`
- (iv) `integer`

**Remark:** Atomic would mean *non divisible*, having no subsets.

#### Question 4

If I execute the expression `x <- 4` in `R` , what is the class of the object '`x`' as determined by the '`class()`' function?

- (i) `vector`
- (ii) `numeric`
- (iii) `complex`
- (iv) `real`

```
x <- 4  
class(x)
```

## Built-In Data Sets

Several data sets , intended as learning tools, are automatically installed when **R** is installed. Many more are installed within packages to complement learning to use those packages. One of these is the famous Iris data set, which is used in many data mining exercises.

- **iris**
- **mtcars**
- **Nile**

To see what data sets are available, simply type **data()**. To load a data set, simply type in the name of the data set. Some data sets are very large. To just see the first few (or last) rows, we use the **head()** function or alternatively the **tail()** function. The default number of rows of these commands is 6. Other numbers can be specified.

```
> head(iris)
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa

```
>
> tail(iris,4)
      Sepal.Length Sepal.Width Petal.Length Petal.Width
147           6.3         2.5         5.0         1.9 v
148           6.5         3.0         5.2         2.0 v
149           6.2         3.4         5.4         2.3 v
150           5.9         3.0         5.1         1.8 v
```

In many situations, it is useful to call a particular data set using the **attach()** command. This will save having to specify the data sets over repeated operations. The file can then be detached using the **detach()** command.

## Other Useful Commands

```
mode(x)
str(x)
dim(x)
length(x)
```

Try out these commands for some other objects, including inbuilt data sets **iris** and **Nile**.

```
iris  
Nile  
Y <- "R"  
Z <- c(TRUE,FALSE,TRUE)
```

### Question 5

What is the class of the object defined by the expression  
`x <- c(4, "a", TRUE)`?

- (i) mixed
- (i) logical
- (i) character
- (i) integer

```
x <- c(4, "a", TRUE)
class(x)
```



### Question 6

If I have two vectors `x <- c(1,3, 5)` and `y <- c(3, 2, 10)`, what is produced by the expression `rbind(x, y)`?

- (i) a vector of length 2
- (ii) a 2 by 3 matrix
- (iii) a vector of length 3
- (iv) a 2 by 2 matrix

```
x <- c(1,3, 5)
y <- c(3, 2, 10)
rbind(x, y)
```

- Use the help file to find out what the commands `rbind()`, `cbind()` and `t()` do.
- The convention is to specify the number of rows, then number of columns.
- The tranpose operator “`t()`” is actually really useful for creating reports from data contained in data frames.

**Question 7**

A key property of vectors in **R** is that

- (i) the length of a vector must be less than 32,768
- (ii) elements of a vector all must be of the same class
- (iii) elements of a vector can only be character or numeric
- (iv) elements of a vector can be of different classes

### Question 8

Suppose I have a list defined as `x <- list(2, "a", "b", TRUE)`.

What does `x[[2]]` give me?

- (i) a list containing the number 2 and the letter "a".
- (ii) a character vector with the elements "a" and "b".
- (iii) a character vector containing the letter "a".
- (iv) a list containing a character vector with the elements "a" and "b".

```
x <- list(2, "a", "b", TRUE)
x[[2]]
```

- Try out `dim()` and `class()` on x also.

### Question 9

Suppose I have a vector  $\mathbf{x} \leftarrow 1:4$  and a vector  $\mathbf{y} \leftarrow 2$ .  
What is produced by the expression  $\mathbf{x} + \mathbf{y}$ ?

- (i) a numeric vector with elements 3, 4, 5, 6.
- (ii) an integer vector with elements 3, 2, 3, 4.
- (iii) a numeric vector with elements 3, 2, 3, 4.
- (iv) an integer vector with elements 3, 2, 3, 6.

### Question 10

Suppose I have a vector

```
x <- c(3, 5, 1, 10, 12, 6)
```

and I want to set all elements of this vector that are less than 6 to be equal to zero. What **R** code achieves this?

- (i) `x[x > 0] <- 6`
- (ii) `x[x < 6] <- 0`
- (iii) `x[x > 6] <- 0`
- (iv) `x[x == 6] <- 0`

This is called **Logical Indexing**. To get an idea of logical indexing, try out the following code snippets.

```
x <- c(3, 5, 1, 10, 12, 6)
x>0
x<6
X==6
```

## The airquality data set

**Some Useful Commands** As well as some of the commands we have seen earlier, it is worth getting to know the following commands also.

1 `head()` and `tail()`

2 `names()`, `rownames()` and `colnames()`

3 `summary()`

4 `complete.cases()`

5 `dim()`, `nrow()` and `ncol()`

Use the `help` command to find out what each of these commands do.

```
help(complete.cases)
```

```
> 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
....
```

## Inspecting the data set

2a) Dimensions

2b) Column names (i.e. variables names)

2c) structure of data frame

Lets compute the dimensions of the data frame *iris*, and also the length of the *Nile* data set.

```
dim(iris)
nrow(iris)
ncol(iris)
length(Nile)
```

Data frames often have specifically named rows and columns. Lets find out the names of the variables (columns) and cases (rows) for the data sets *iris* and *mtcars*.

```
names(iris)
colnames(iris)
rownames(iris) # simply the case numbers.

names(mtcars)
```

```
rownames(mtcars)  
colnames(mtcars)
```



## The `summary()` command

The `summary()` command can be used to extract a short statistical summary (if applicable) from each column of the data frame. If there are missing values, the frequency of missing values will also be listed for each column.

```
> summary(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.400
Median :5.800	Median :3.000	Median :4.350	Median :1.300
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500

## Types of Data in Dataframes

What is the data type of each column in the iris data set?  
To find out, we use the `str()` command.

```
str(iris)
```

The output of this command is given below. There are four numeric variables, and one factor (i.e. categorical) variable.

```
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.2
 $ Species      : Factor w/ 3 levels "setosa","versicol
>
```

## The head() and tail() function

The head and tail functions can be used to access the first six and last six rows of a data frame. If a different number of rows is required, all you have to do is specify that number as an additional argument.

```
head(iris)      #First 6 rows
head(iris,2)    #First 2 rows

tail(iris)      #Last 6 rows
tail(iris,4)    #Last 4 rows
```

## Accessing a particular row or set of rows

- Each value in a dataframe can be accessed directly by specifying the row and column i.e. `df[r,c]`.
- To access a particular row, simply specify the row number, while leaving the column number blank i.e. `df[r,]`
- To access a particular column, simple specify the column number, while leaving the row number blank i.e. `df[,c]`

```
iris[10,2]
```

```
iris[10,]
```

```
Formaldehyde[,2]
```

```
> iris[10,2]
```

```
[1] 3.1
```

```
> iris[10,]
```

```
      Sepal.Length Sepal.Width Petal.Length Petal.Width Species
10           4.9           3.1           1.5           0.1  setosa
```

```
>
```

```
> Formaldehyde[,2]
[1] 0.086 0.269 0.446 0.538 0.626 0.782
```

## Missing data

- Determining the number of missing data items
- Performing statistical operations removing missing data

As stated previously, the **summary()** command can be used to determine the number of missing data items in a data frame. The additional argument **na.rm=TRUE** can also be used with certain functions (see the help file)

```
> X <- c(4,6,3,12,NA,8)
> mean(X)
[1] NA
>
> summary(X)
      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
      3.0     4.0     6.0     6.6     8.0    12.0      1
>
> mean(X ,na.rm = TRUE)
[1] 6.6
```

## Subsetting Data

### Logical and Relational Operator

- AND - The logical operator is &
- OR - The logical operator is ||

### Selection using the subset() Function

The **subset()** function is the easiest way to select variables and observation.

#### Example 1

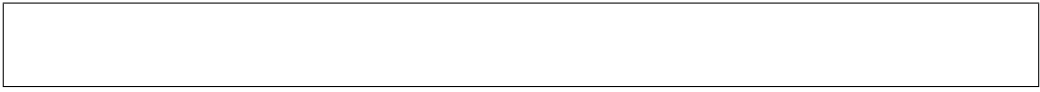
In the following example we will use the *iris* data set, we select all rows that have a value of sepal length of 6 or more, and determine how many observations there are, and then compute the mean of the petal lengths. (The answer is 5.263, from the summary output).

```
#call the subset iris.2

iris.2 = subset(iris,iris$Sepal.Length >= 6)

dim(iris.2)

summary(iris.2)
```



## Example 2

There are three types of iris - setosa, versicolour and virginica. Suppose we wish to compute the median of petal widths for the setosa irises only.

The equality operator is `==`.

```
> iris.setosa =subset(iris,iris$Species=="setosa")
> summary(iris.setosa)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Min. :4.300	Min. :2.300	Min. :1.000	Min. :0.100
1st Qu.:4.800	1st Qu.:3.200	1st Qu.:1.400	1st Qu.:0.200
Median :5.000	Median :3.400	Median :1.500	Median :0.300
Mean :5.006	Mean :3.428	Mean :1.462	Mean :0.376
3rd Qu.:5.200	3rd Qu.:3.675	3rd Qu.:1.575	3rd Qu.:0.400
Max. :5.800	Max. :4.400	Max. :1.900	Max. :0.500

Species

setosa :50

versicolor: 0

virginica : 0

## Example 3

In the following example we will use the *iris* data set, we select all rows that have a value of sepal length of 6 or more, but have sepal width is at least 2.5.

```
> iris.3 = subset(iris,(iris$Sepal.Length >= 6)&(iris$Sepal.Width >= 2.5))
> summary(iris.3)
```



Sepal.Length	Sepal.Width	Petal.Length	Petal
Min. :6.000	Min. :2.500	Min. :4.000	Min.
1st Qu.:6.300	1st Qu.:2.800	1st Qu.:4.750	1st Qu.
Median :6.500	Median :3.000	Median :5.300	Median
Mean :6.641	Mean :3.013	Mean :5.313	Mean
3rd Qu.:6.900	3rd Qu.:3.200	3rd Qu.:5.750	3rd Qu.
Max. :7.900	Max. :3.800	Max. :6.900	Max.

Species

setosa : 0

versicolor:21

virginica :42

**Question 11**

In the dataset provided for this Quiz, what are the column names of the dataset?

- (i) Ozone, Solar.R, Wind, Temp, Month, Day
- (ii) Ozone, Solar.R, Wind
- (iii) Month, Day, Temp, Wind
- (iv) 1, 2, 3, 4, 5, 6

## Sequences and Numerical Indexing

A sequence of integers can be created using the colon symbol. The sequence may either be *count-up* or *count-down*.

Importantly, the sequence will return all integers between the upper and lower bound, and including both the upper and lower bound (*Other languages can be different in this respect*)

```
0:5  
1:6  
-4:5  
10:1
```

A contiguous group of rows from a data frame may be extracted using the appropriate sequence of values as indices. Likewise for a contiguous group of columns

```
iris[1:6,]      # First Six Rows  
iris[,3:4]      # Third and Fourth Columns  
iris[1:40,2:3]  # 40 Rows, Columns 2 and 3
```

# 1 Indexing and Subsetting

## 1.1 Relational and Logical Operators

Relational operators allow for the comparison of values in vectors.

greater than	>
less than	<
equal to	==
less than or equal to	<=
greater than or equal to	>=
not equal to	!=

- Note the difference of the equality operator "==" with assignment operator "=".
- & and && indicate logical AND and || and ||| indicate logical OR.
- The shorter form performs element-wise comparisons in much the same way as arithmetic operators. The longer form is appropriate for programming control-flow and typically preferred in "if" clauses.
- We can use relational operators to subset vectors (as well as more complex data objects such as data frames, which we will meet later).

- We specify the relational condition in square brackets.
- We can construct compound relational conditions too, using logical operators

```
> vec=1:19
> vec[vec<5]
[1] 1 2 3 4
> vec[(vec<6)|(vec>16)]
[1] 1 2 3 4 5 17 18 19
```

## Question 12

Extract the first 2 rows of the data frame and print them to the console. What does the output look like?

```
Ozone Solar.R Wind Temp Month Day
1     41     190  7.4   67     5   1
2     36     118  8.0   72     5   2
```

```
Ozone Solar.R Wind Temp Month Day
1      7      NA  6.9   74     5  11
2     35     274 10.3   82     7  17
```

```
Ozone Solar.R Wind Temp Month Day
1     18     224 13.8   67     9  17
2     NA     258  9.7   81     7  22
```

```
Ozone Solar.R Wind Temp Month Day
1      9      24 10.9   71     9  14
2     18     131  8.0   76     9  29
```

### Question 13

How many observations (i.e. rows) are in this data frame?

- 129
- 153
- 160
- 45

### Question 14

Extract the last 2 rows of the data frame and print them to the console. What does the output look like?

```
Ozone Solar.R Wind Temp Month Day
152    11     44  9.7   62     5  20
153   108    223  8.0   85     7  25
```

```
Ozone Solar.R Wind Temp Month Day
152    34    307 12.0   66     5  17
153    13     27 10.3   76     9  18
```

```
Ozone Solar.R Wind Temp Month Day
152    18    131  8.0   76     9  29
153    20    223 11.5   68     9  30
```

```
Ozone Solar.R Wind Temp Month Day
152    31    244 10.9   78     8  19
153    29    127  9.7   82     6   7
```



**Question 15**

What is the value of Ozone in the 47th row?

- (i) 63
- (ii) 34
- (iii) 18
- (iv) 21

### Question 16

How many missing values (**NAs**) are in the *Ozone* column of this data frame?

- (i) 37
- (ii) 78
- (iii) 43
- (iv) 9

```
names(airquality)
attach(airquality)
is.na(Ozone)

!is.na(Ozone)

which(is.na(Ozone))

length(which(is.na(Ozone)))
detach(airquality)
```

**Question 17**

What is the mean of the Ozone column in this dataset? Exclude missing values (coded as NA) from this calculation.

- (i) 42.1
- (ii) 18.0
- (iii) 53.2
- (iv) 31.5

**Question 18**

Extract the subset of rows of the data frame where Ozone values are above 31 and Temp values are above 90.

What is the mean of Solar.R in this subset?

- (i) 185.9
- (ii) 212.8
- (iii) 334.0
- (iv) 205.0

**Question 19**

What is the mean of "Temp" when "Month" is equal to 6?

- (i) 90.2
- (ii) 85.6
- (iii) 79.1
- (iv) 75.3

**Question 20**

What was the maximum ozone value in the month of May (i.e. Month = 5)?

- (i) 100
- (ii) 115
- (iii) 18
- (iv) 97