BT1101: Introduction to Business Analytics
Tutorial 2

R Programming

Basics

STRUCTURE OF TUTORIALS

Duration:

45 mins

Content:

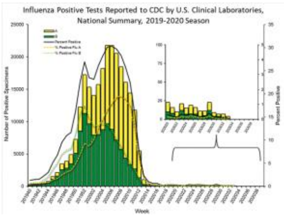
- Cover previous week’s tutorial assignment
- Basic functions in R
- Atomic datatypes in R
- Data structures in R

TUTORIAL 1 DISCUSSION

QUESTION 1A

Based on the news articles, what was the data the researchers with the Scripps Research Translational Institute collected in their study?

- Resting heart rate
- Sleep patterns
- Activity levels



CDC
CENTERS FOR DISEASE CONTROL AND PREVENTION

- Weekly estimates of influenza like illness from the CDC (state level).

What type of analytics did they perform with the data?

- **Predictive analytics** was employed by the study to predict flu occurrence based on heart rate, sleep measures and daily activities (at state level).
- **Descriptive analytics:** understand the demographics. E.g., average age of users is 43 years and 60% of them are female.

Raw data collected by Fitbit vs. data the researchers obtained.

- Data collected by Fitbit is used to estimate the metrics that the researchers obtained.
- Data collected by Fitbit includes measures e.g., volume changes in the capillaries above the wrist that can be used to compute the metrics.

QUESTION 1B

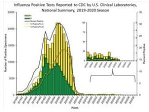
How does the approach taken by this research team add value to current methods of flu outbreak detection and response?

Current Methods & their limitations

01

- Overestimation of flu incidents.
- Teasing apart actual illness from interest.

02



- Often falls a few weeks behind the actual outbreak.

The Employed Approach & its Value

01

- More precise detections.

02

- help enact timely outbreak response measures.

QUESTION 1C

What are the limitations of their approach? What challenges might they face in implementing their approach of flu outbreak detection and prediction?

Accuracy of wearables

METRIC

Not specific to flu

Selection Bias

Privacy concerns

Capital intensive

User training

Inconsistent user behavior

BT1101: Introduction to Business Analytics
Tutorial 2

R Programming

Basics

INTRODUCTION TO DATA STRUCTURES IN R

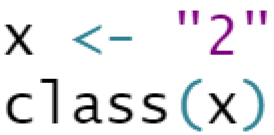
- **Vector**
  - Vectors can carry 1 datatype e.g., numeric, character or logical
  - y <- c(20,36,10,10, 10)
  - Size <- c("medium", "small", "big", "big")
- **Matrix**
  - A collection of numbers arranged into a fixed number of rows and columns.
  - mat1 <- matrix(1:4, nrow = 2, ncol = 2)
- **Array**
  - Multi-dimensional Data structures. In an array, data is stored in the form of matrices, row, and as well as in columns.
- **Lists**
  - Can contain elements of different data types – like strings, numbers, vectors and another list.
  - out_list <- list(vec, char_vec, logic_vec)
- **Data frames**
  - Tabular data
  - data_frame <- data.frame(int_vec, char_vec, bool_vec)

INTRODUCTION TO DATA TYPES IN R

- A basic concept in (statistical) programming is called a **variable**.
- A variable allows you to store a value (e.g. “2”) or an object (e.g. a function description) in R.
- You can then later use this variable's name to easily access the value or the object that is stored within this variable.
- Every variable has a **class: data type.**
  - Numeric
  - Integers
  - Logical
  - Characters
  - Factors

QUESTION 1A

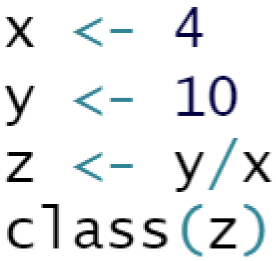
i



```
x <- "2"
class(x)
```

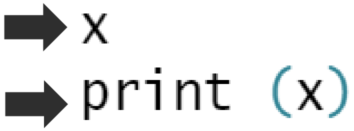
character

ii



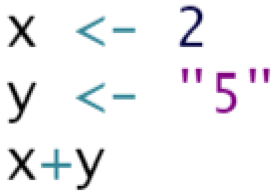
```
x <- 4
y <- 10
z <- y/x
class(z)
```

numeric

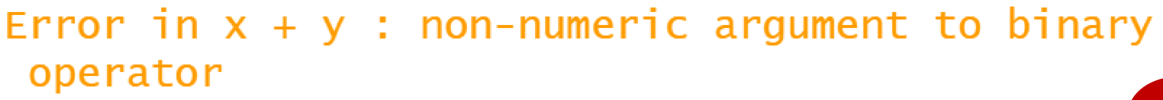


```
x
print (x)
```

You ran the chunks of code in the order i & ii. What’s the value of x?



```
x <- 2
y <- "5"
x+y
```



```
Error in x + y : non-numeric argument to binary
 operator
```

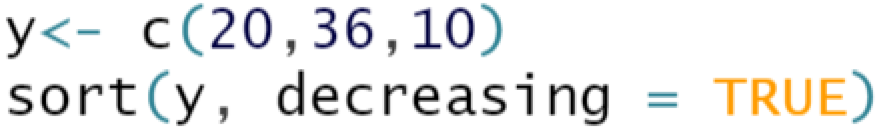
!

Why?

QUESTION 1B

- **Sort**: values in ascending or descending fashion
- **Order**: index/position of values. R index starts at 1 NOT 0

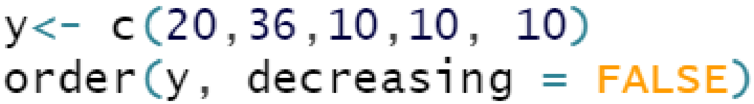
i



```
y<- c(20,36,10)
sort(y, decreasing = TRUE)
```

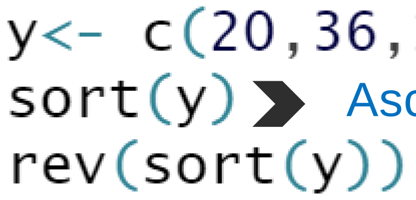
36 20 10

ii



```
y<- c(20,36,10,10, 10)
order(y, decreasing = FALSE)
```

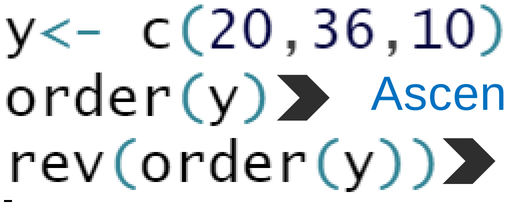
3 4 5 1 2



```
y<- c(20,36,10)
sort(y)
rev(sort(y))
```

Ascending

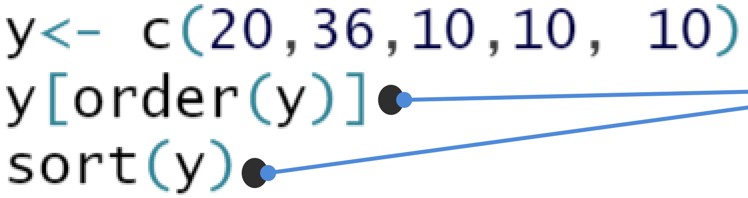
Descending



```
y<- c(20,36,10)
order(y)
rev(order(y))
```

Ascending

Descending



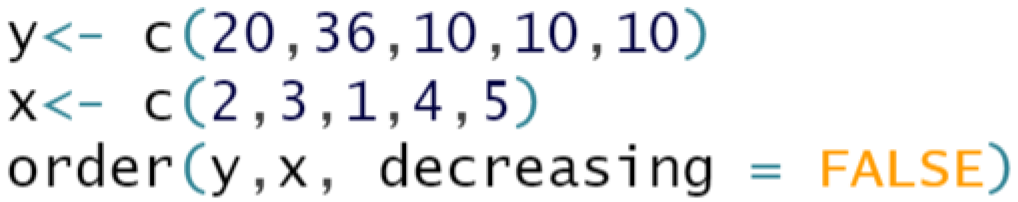
```
y<- c(20,36,10,10, 10)
y[order(y)]
sort(y)
```

Same output

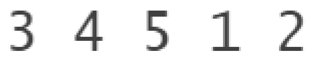
order will be useful when you want to sort vectors in a dataframe based on a specific variable.

QUESTION 1B

iii

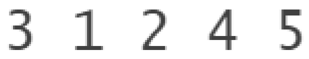


```
y<- c(20,36,10,10,10)
x<- c(2,3,1,4,5)
order(y,x, decreasing = FALSE)
```



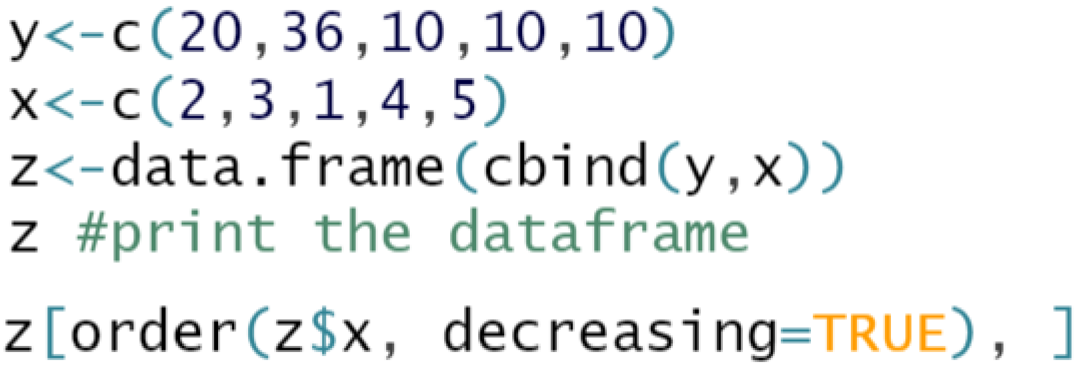
```
3 4 5 1 2
```

What if: order(x,y…)



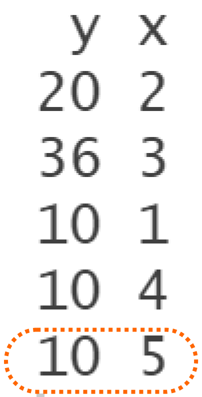
```
3 1 2 4 5
```

iv



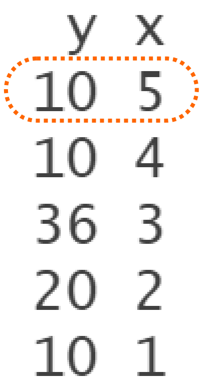
```
y<-c(20,36,10,10,10)
x<-c(2,3,1,4,5)
z<-data.frame(cbind(y,x))
z #print the dataframe

z[order(z$x, decreasing=TRUE), ]
```

Before order



```
 y x
20 2
36 3
10 1
10 4
10 5
```

After order

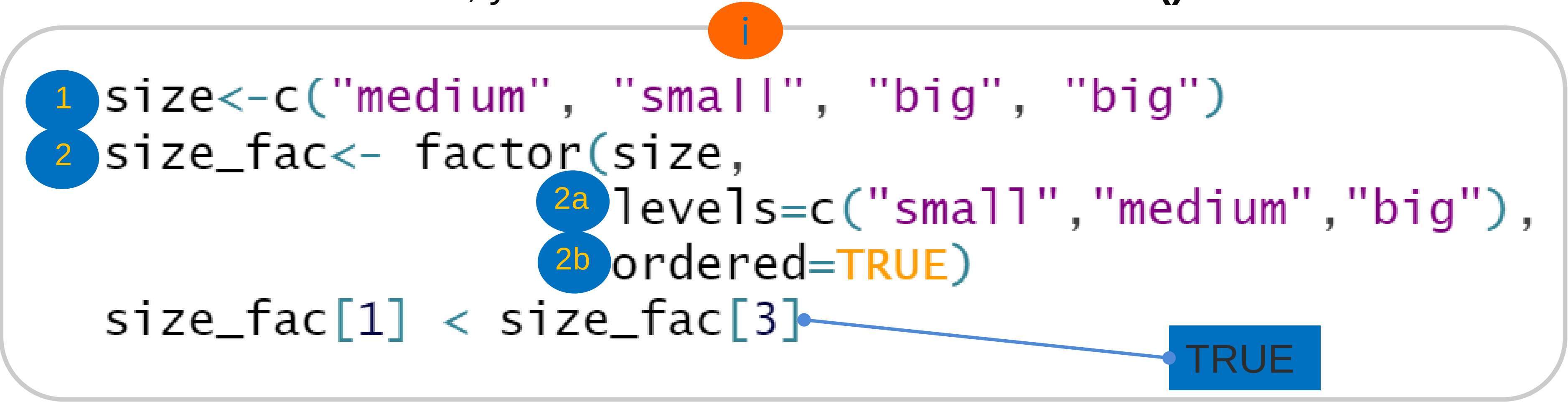


```
 y x
10 5
10 4
36 3
20 2
10 1
```

Sort every row by x

QUESTION 1C

- Factors store **categorical variables**
- Categorical variables can be **ordinal** or **nominal**
- To create factors in R, you make use of the **function factor()**



```
size<-c("medium", "small", "big", "big")
size_fac<- factor(size,
                  levels=c("small","medium","big"),
                  ordered=TRUE)
size_fac[1] < size_fac[3]
```

```
TRUE
```

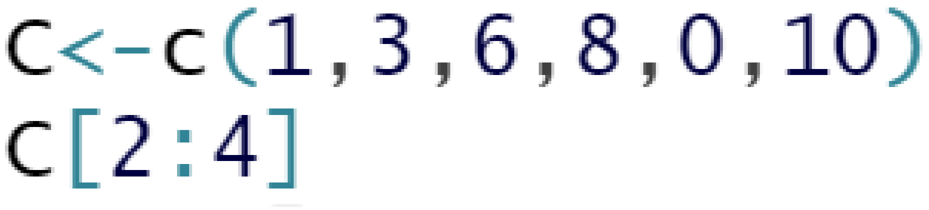
1 First thing that you have to do is create a vector that contains all the observations that belong to a limited number of categories.

2 Create factor

- define levels; R will take your inscription of levels. It will not auto-discern!
- Specify if the factor is ordinal

QUESTION 1D

i

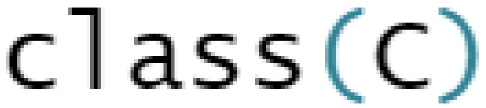


```
C<-c(1,3,6,8,0,10)
c[2:4]
```

3 6 8

Select elements 2 to 4 from vector C

ii



```
class(C)
```

numeric

QUESTION 1E

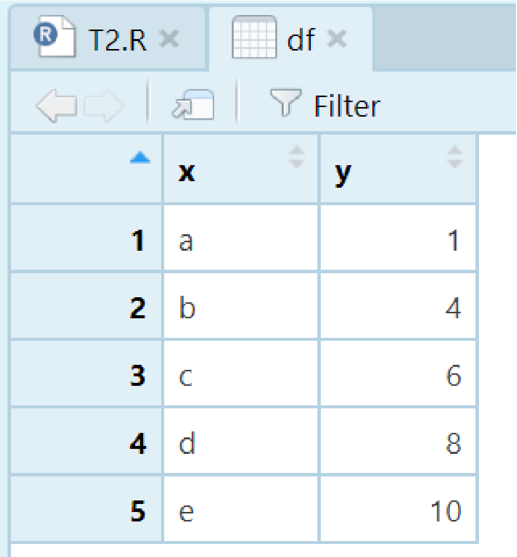
character vector

numeric vector

```
df<-data.frame(x=c("a","b","c","d","e"), y=c(1,4,6,8,10),
                stringsAsFactors = FALSE)
View(df)
```

The argument '**stringsAsFactors**' is an argument to the 'data. frame()' function in **R**. It is a logical that indicates whether strings in a data frame should be treated as factor variables or as just plain strings

```
View(df)
```



	x	y
1	a	1
2	b	4
3	c	6
4	d	8
5	e	10

The View function invokes a spreadsheet-style data viewer on a matrix-like R object.

QUESTION 1E

i

```
class(df$x)
```

character

ii

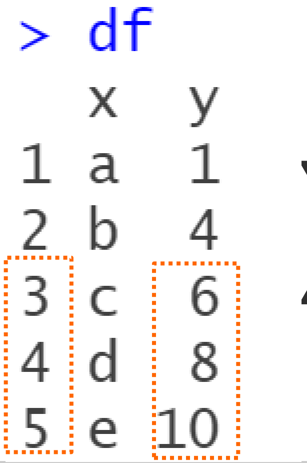
```
class(df$y)
```

numeric

iii

```
df[c(3:5),"y"]
```

For variable y, get values at index 3-5.
NB: Index starts at 1



```
> df
  x  y
1 a  1
2 b  4
3 c  6
4 d  8
5 e 10
```

```
6 8 10
```

iv

```
df$y<-as.integer(df$y)
class(df$y)
```

integer

v

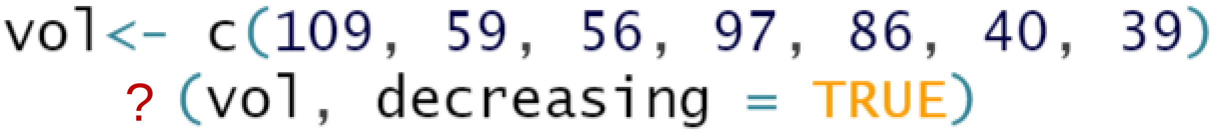
```
subset(df,y>6,
          select=x)
```

```
  x
4 d
5 e
```

Select x values where y is >6

QUESTION 2A: WHAT’S THE MISSING CODE?

i



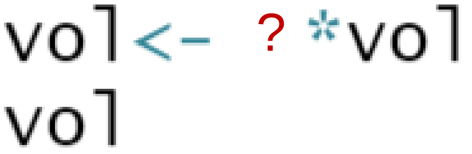
```
vol<- c(109, 59, 56, 97, 86, 40, 39)
    ? (vol, decreasing = TRUE)
```

output

```
1   4   5   2   3   6   7
```

order

ii



```
vol<- ? *vol
vol
```

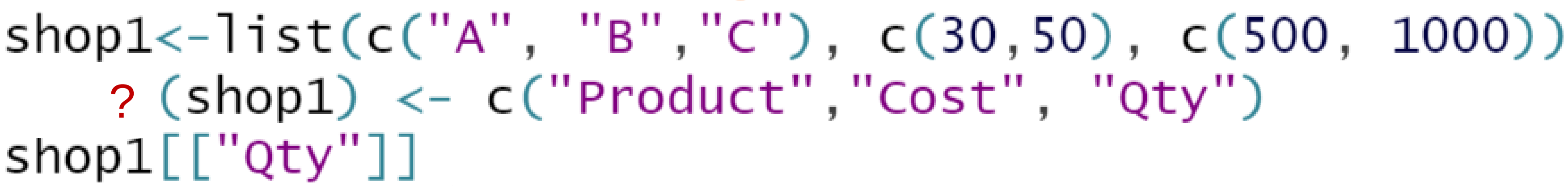
output

```
218   118   112   194   172   80   78
```

2

QUESTION 2B: WHAT'S THE MISSING CODE?

i



```
shop1<-list(c("A", "B","C"), c(30,50), c(500, 1000))
    ? (shop1) <- c("Product","Cost", "Qty")
shop1[["Qty"]]
```

output

500   1000

- names
- names: Functions to get or set the names of an object.
- shop1[["Qty"]] ➤ retrieves a list from another list

ii

```
shop1$ ?
```

output

"A" "B" "C"

shop1$Product

iii

```
shop1$ ?
```

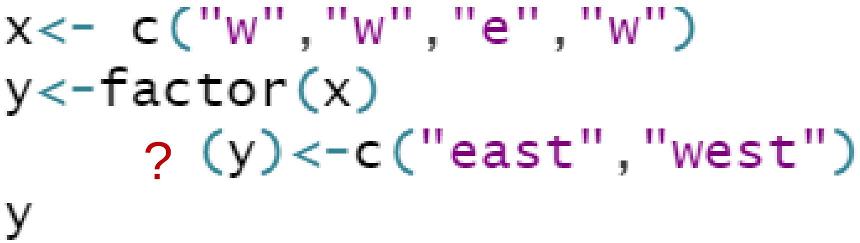
output

30

shop1$Cost[1]

QUESTION 2C: WHAT’S THE MISSING CODE?

i



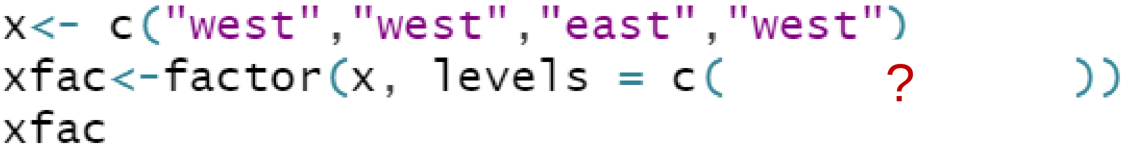
```
x<- c("w","w","e","w")
y<-factor(x)
      ? (y)<-c("east","west")
y
```

output

```
[1] west west east west
Levels: east west
```

levels

ii



```
x<- c("west","west","east","west")
xfac<-factor(x, levels = c(      ?      ))
xfac
```

output

```
[1] west west east west
Levels: east west
```

"east","west"

QUESTION 2D: WHAT'S THE MISSING CODE?

```
Candidates <- c("Mary","Natalie","James","Pete")
Vote <- c(23, 44, 5, 66)
```

i

```
Vote[  ?  ]
```

output

```
[1] 5
```

Vote[c(3)]

ii

```
Candidates[   ?   ]
```

output

```
[1] "Mary" "Pete"
```

Candidates[c(1,4)]

iii

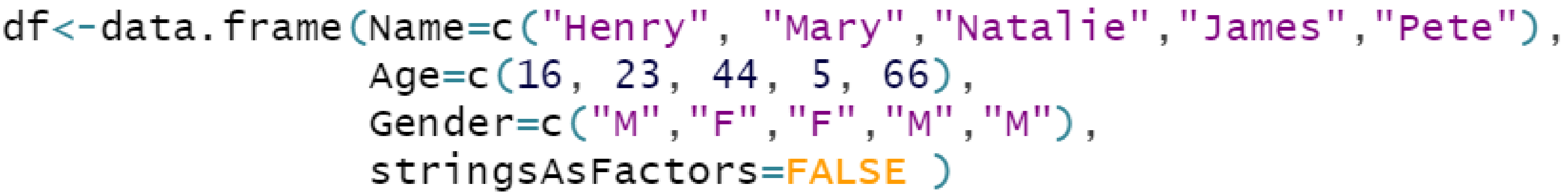
```
dfvoting <-    ?   (Candidates,Vote)
dfvoting
```

output

```
     Candidates Vote
[1,] "Mary"     "23"
[2,] "Natalie"  "44"
[3,] "James"    "5"
[4,] "Pete"     "66"
```

cbind

QUESTION 2E: WHAT'S THE MISSING CODE?

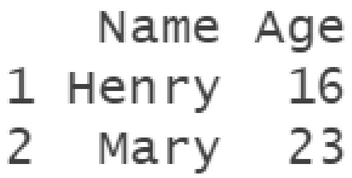


```
df<-data.frame(Name=c("Henry", "Mary","Natalie","James","Pete"),
               Age=c(16, 23, 44, 5, 66),
               Gender=c("M","F","F","M","M"),
               stringsAsFactors=FALSE )
```

i

```
df[        ?        ]
```

output



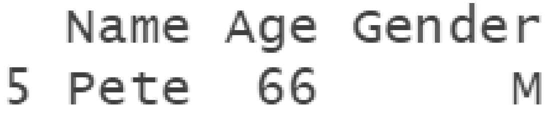
```
   Name Age
1 Henry  16
2  Mary  23
```

df[c(1,2), c(1,2)]

ii

```
? (df,Age>50)
```

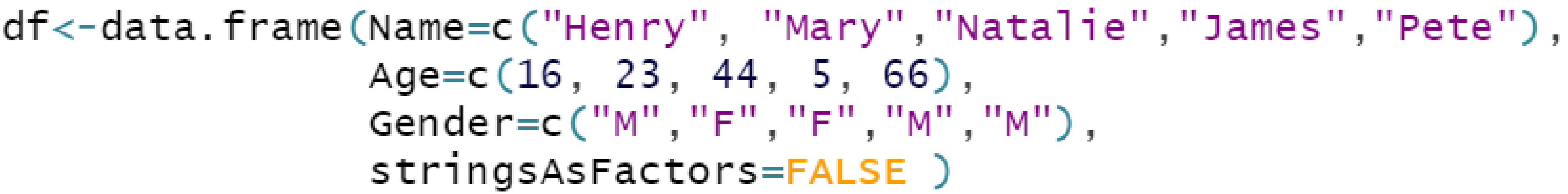
output



```
  Name Age Gender
5 Pete  66      M
```

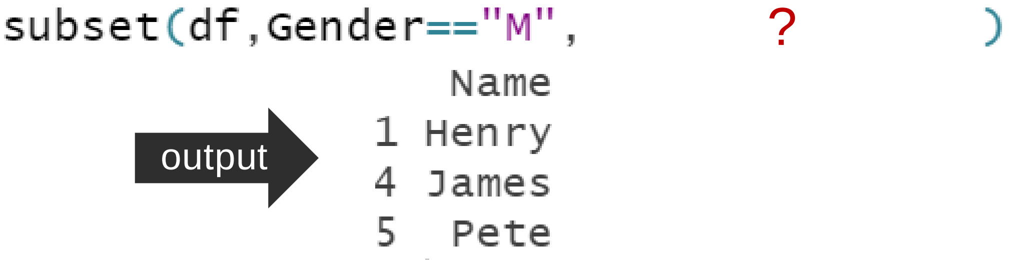
subset(df,Age>50)

QUESTION 2E: WHAT’S THE MISSING CODE?



```
df<-data.frame(Name=c("Henry", "Mary","Natalie","James","Pete"),
               Age=c(16, 23, 44, 5, 66),
               Gender=c("M","F","F","M","M"),
               stringsAsFactors=FALSE )
```

iii



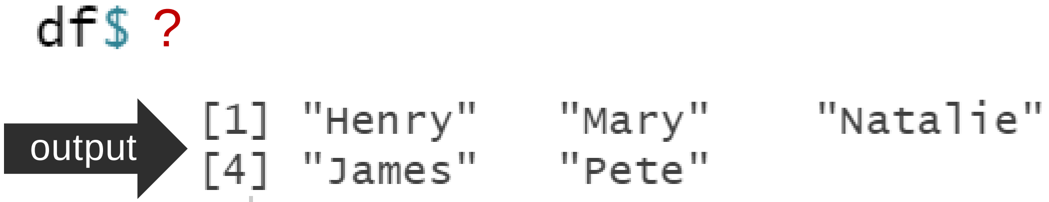
```
subset(df,Gender=="M",          ?          )
```

output

```
   Name
1 Henry
4 James
5  Pete
```

subset(df,Gender=="M",select = "Name")

iv



```
df$ ?
```

output

```
[1] "Henry"   "Mary"     "Natalie"
[4] "James"   "Pete"
```

df$Name

QUESTION 3

A variable *rain_vol* contains the following values (which is the rain volume for each day): 100, 150, 140, 125, 20, 30, 55

What is the code to create the *rain_vol* vector?

```
rain_vol<-c(100, 150,140,125,20,30,55)
```

What is the code to assign the first 3 letters of the days of the week (from “Mon”, “Tue”… “Sun”) as names of the *rain_vol* vector?

```
names(rain_vol) <- c("Mon","Tue","Wed","Thu","Fri","Sat","Sun")
```

What is the code to sort *rain_vol* in increasing volume?

```
sort(rain_vol,decreasing=FALSE)
```

There was an error in the measuring gauge. Could you subtract 10 from each of the values in the *rain_vol*? What is the code to do this?

```
rain_vol<-rain_vol-10
rain_vol
```

THANK YOU. SEE YOU NEXT WEEK.