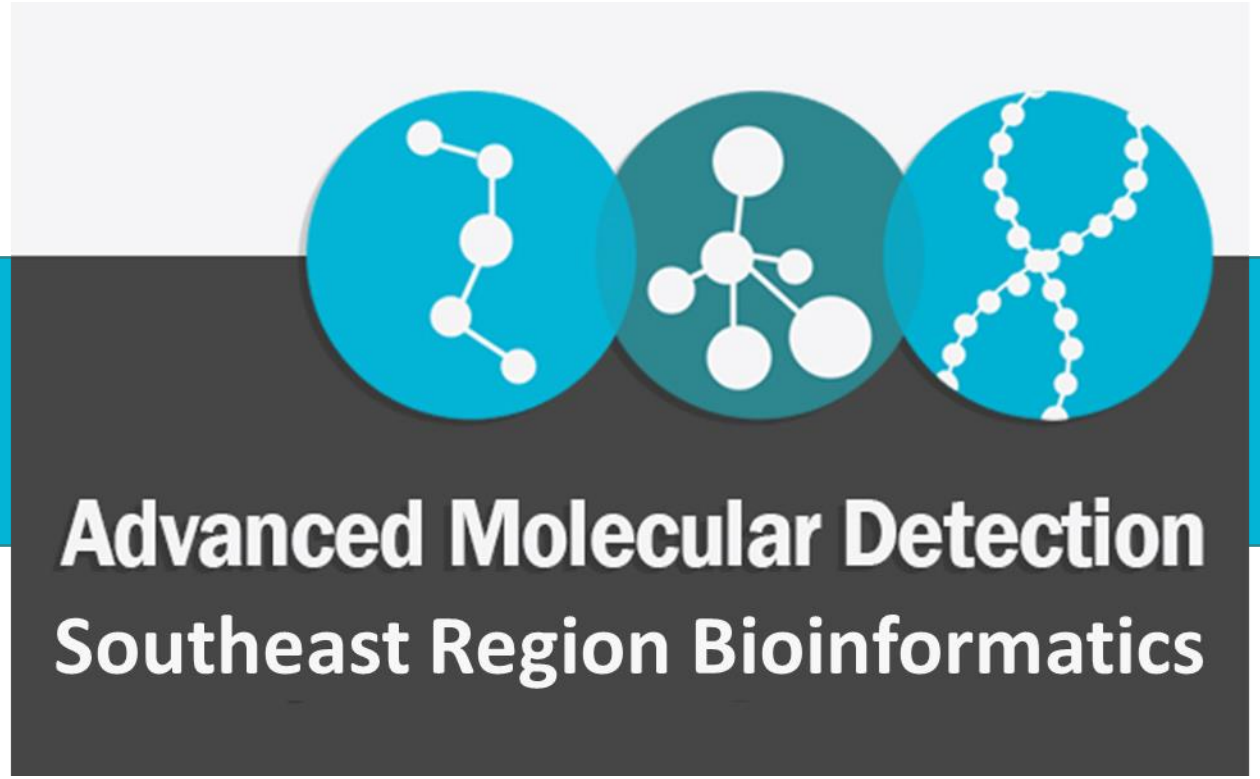


Data Types & Structures

05/13/2024

Outline

Agenda

Notes

Data Types in R

Objects in R

Questions

Agenda

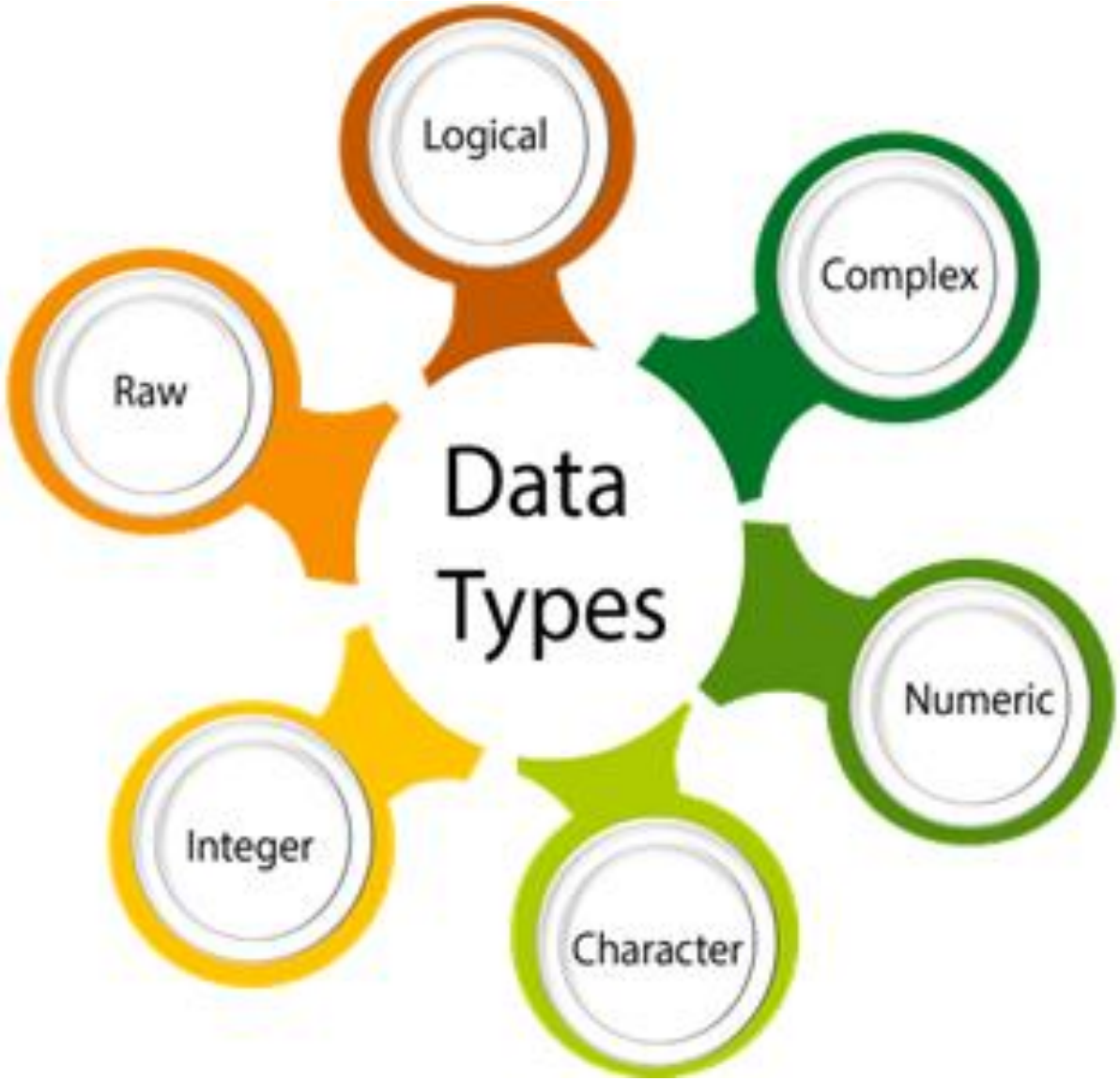
June 3 – Importing Data & Iterations in R Part - 3
June 17 – Graphics in R Part - 4

Future Trainings
- ONT & FL's Flisochar pipeline
- StaPH-B Toolkit Programs/Pipelines
- GISAID flagged SARS-CoV-2
- R Training Series
- Dryad pipeline
- …and more

Advanced Molecular Detection
Southeast Region Bioinformatics

R Data Types

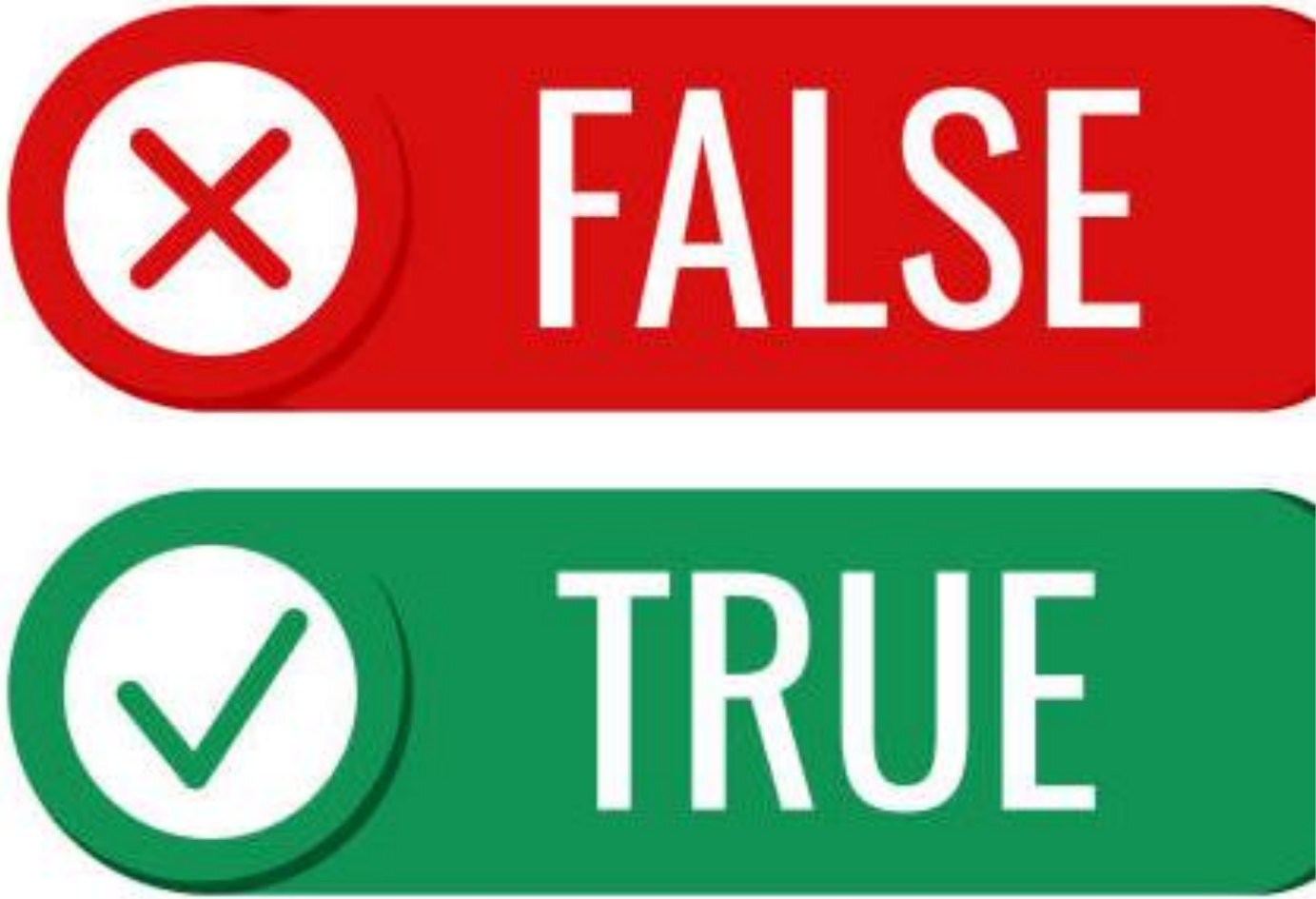
- A variable can store different types of values such as numbers, characters, etc.
- Different types of data that can be used in your code are called data types
- Data types include:
    - Logical
    - Numeric
    - Integer
    - Complex
    - Character
    - Raw



https://www.javatpoint.com/r-data-types

Logical Data Type

- Logical data type in R is also known as boolean data type
- It can only have two values: True and False



Advanced Molecular Detection
Southeast Region Bioinformatics

Logical Data Type

Input:

```
1  x <- TRUE
2  print(x)
3  class(x)
4
5  y <- FALSE
6  print(y)
7  print(class(y))
```

Output:

```
> x <- TRUE
> print(x)
[1] TRUE
> class(x)
[1] "logical"
>
> y <- FALSE
> print(y)
[1] FALSE
> print(class(y))
[1] "logical"
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Numeric Data Type

- Numeric data type represents all real numbers with or without decimal values

Numeric Data Type

Input:

```
 9  # float values
10  weight <- 73.5
11  print(weight)
12  print(class(weight))
13  # real numbers
14  height <- 183
15  print(height)
16  print(class(height))
```

Output:

```
> # float values
> weight <- 73.5
> print(weight)
[1] 73.5
> print(class(weight))
[1] "numeric"
> # real numbers
> height <- 183
> print(height)
[1] 183
> print(class(height))
[1] "numeric"
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Integer Data Type

- Integer data type specifies real values without decimal points
- Here, suffix L is used to specify integer data

Integer Data Type

Input:

```
18  integer_variable <- 257L
19  print(class(integer_variable))
```

Output:

```
> integer_variable <- 257L
> print(class(integer_variable))
[1] "integer"
```

Complex Data Type

- Complex vectors can be created with complex. The vector can be specified either by giving its length, its real and imaginary parts, or modulus and argument
- Complex data type is used to specify purely imaginary values in R. Here suffix i is used to specify the imaginary part

Complex Data Type

Input:

```
21  complex_value <- 3 + 2i
22  print(class(complex_value))
```

Output:

```
> complex_value <- 3 + 2i
> print(class(complex_value))
[1] "complex"
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Character Data Type

- Character data type is used to specify character or string values in a variable
- String is a set of characters. For example, ‘A’ is a single character and “Apple” is a string

Character Data Type

Input:

```
24  #create a string variable
25  fruit <- "Apple"
26  print(class(fruit))
27
28  #create a character variable
29  my_char <- 'A'
30  print(class(my_char))
```

Output:

```
> #create a string variable
> fruit <- "Apple"
> print(class(fruit))
[1] "character"
>
> #create a character variable
> my_char <- 'A'
> print(class(my_char))
[1] "character"
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Raw Data Type

- A raw data type specifies values as raw bytes
- charToRaw() – converts character data to raw data
- rawToChar() – converts raw data to character data

Raw Data Type

Input:

```
33  # convert character to raw
34  raw_variable <- charToRaw("Welcome to training")
35
36  print(raw_variable)
37  print(class(raw_variable))
38
39  # convert raw to character
40  char_variable <- rawToChar(raw_variable)
41
42  print(char_variable)
43  print(class(char_variable))
```

Output:

```
> # convert character to raw
> raw_variable <- charToRaw("Welcome to training")
>
> print(raw_variable)
 [1] 57 65 6c 63 6f 6d 65 20 74 6f 20 74 72 61 69 6e 69 6e 67
> print(class(raw_variable))
[1] "raw"
>
> # convert raw to character
> char_variable <- rawToChar(raw_variable)
>
> print(char_variable)
[1] "Welcome to training"
> print(class(char_variable))
[1] "character"
```

Advanced Molecular Detection
Southeast Region Bioinformatics

R Objects

- R objects are simply a collection of variables and functions
- Types of objects include:
  - Vectors
  - Lists
  - Matrices
  - Factors
  - Data frames

Advanced Molecular Detection
Southeast Region Bioinformatics

Vectors

- Basic type of objects
- Stores homogenous data types such as character, doubles, integers, raw, logical, and complex

Vectors

Input:

```
45  #create vectors
46  x <- c(1,2,3,4)
47  y <- c("a", "b", "c", "d")
48  z <- 5
49  print(x)
50  print(class(x))
51  print(y)
52  print(class(y))
53  print(z)
54  print(class(z))
55  print(y[3])
```

Output:

```
> #create vectors
> x <- c(1,2,3,4)
> y <- c("a", "b", "c", "d")
> z <- 5
> print(x)
[1] 1 2 3 4
> print(class(x))
[1] "numeric"
> print(y)
[1] "a" "b" "c" "d"
> print(class(y))
[1] "character" .
> print(z)
[1] 5
> print(class(z))
[1] "numeric"
> print(y[3])
[1] "c"
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Lists

- List is a collection of similar or different types of data
- list() function is used to create a list

Lists

Input:

```
57  #list with similar type of data
58  list1 <- list(56, 74, 83, 92)
59  #list with different type of data
60  list2 <- list("Elsa", TRUE, 45)
61  #access elements using indexing
62  print(list2[3])
63  #add items to the list
64  append(list2, "Cathy")
65  # modify a list element
66  list2[2] <- "Anna"
67  print(list2)
68  #remove items from list
69  print(list2[-2])
```

Output:

```
> # modify a list element
> list2[2] <- "Anna"
> print(list2)
[[1]]
[1] "Elsa"

[[2]]
[1] "Anna"

[[3]]
[1] 45

> #remove items from list
> print(list2[-2])
[[1]]
[1] "Elsa"

[[2]]
[1] 45
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Data Frame

- A data frame is a two-dimensional data structure which can store data in tabular format
- Data frames have rows and columns, and each column can be a different vector and different vectors can be of different data types

Data Frame

Input:

```
70  #create a data frame
71  dataframe1 <- data.frame (
72    Name = c("John", "Anna", "Elsa"),
73    Age = c(15, 22, 17),
74    Vote = c(TRUE, FALSE, TRUE)
75  )
76  print(dataframe1)
```

Output:

```
> #create a data frame
> dataframe1 <- data.frame (
+   Name = c("John", "Anna", "Elsa"),
+   Age = c(15,'22, 17),
+   Vote = c(TRUE, FALSE, TRUE)
+ )
> print(dataframe1)
  Name Age  Vote
1 John  15  TRUE
2 Anna  22 FALSE
3 Elsa  17  TRUE
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Matrix

- Matrix is a two-dimensional data structure where data are arranged into rows and columns
- Syntax of the matrix() function is
    - Vector - the data items of same type
    - nrow - number of rows
    - ncol - number of columns
    - byrow(optional) – if TRUE, the matrix is filled row-wise. By default, the matrix is filled column-wise

Creating a Matrix

Creating a 2 by 3 matrix

Input:

```
78  # create a 2 by 3 matrix
79  matrix1 <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3, byrow = TRUE)
80  print(matrix1)
```

Output:

```
> # create a 2 by 3 matrix
> matrix1 <- matrix(c(1,2,3,4,5,6), nrow = 2, ncol = 3, byrow = TRUE)
> print(matrix1)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Combine Two Matrices in R

Input:

```
82  # create two 2 by 2 matrices
83  even_numbers <- matrix(c(2,4,6,8), nrow = 2, ncol =2)
84  odd_numbers <- matrix(c(1,3,5,7), nrow = 2, ncol = 2)
85
86  # combine two matrices by column
87  total1 <- cbind(even_numbers, odd_numbers)
88  print(total1)
89
90  # combine twp matrices by row
91  total2 <- rbind(even_numbers, odd_numbers)
92  print(total2)
```

Output:

```
> # create two 2 by 2 matrices
> even_numbers <- matrix(c(2,4,6,8), nrow = 2, ncol =2)
> odd_numbers <- matrix(c(1,3,5,7), nrow = 2, ncol = 2)
>
> # combine two matrices by column
> total1 <- cbind(even_numbers, odd_numbers)
> print(total1)
     [,1] [,2] [,3] [,4]
[1,]    2    6    1    5
[2,]    4    8    3    7
>
> # combine twp matrices by row
> total2 <- rbind(even_numbers, odd_numbers)
> print(total2)
     [,1] [,2]
[1,]    2    6
[2,]    4    8
[3,]    1    5
[4,]    3    7
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Arrays

- An array is a data structure that stores data of the same type in more than two dimensions
- Difference between vectors, matrices, and arrays are
    - Vectors are uni-dimensional arrays
    - Matrices are two-dimensional arrays
    - Arrays can have more than two dimension
- Syntax of the array() function is

```
array(vector, dim = c(nrow, ncol, nmat))
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Create an Array in R

Input:

```
94  # Array, create a two 2 by 3 matrix
95  array1 <- array(c(1:12), dim = c(2,3,2))
96  print(array1)
```

Output:

```
> # Array, create a two 2 by 3 matrix
> array1 <- array(c(1:12), dim = c(2,3,2))
> print(array1)
, , 1

     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

, , 2

     [,1] [,2] [,3]
[1,]    7    9   11
[2,]    8   10   12
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Factors in R

- A factor is a data structure that is used to work with categorizable data
- Factor takes vector as an argument
- Syntax for creating a factor is

```
factor(vector)
```

Advanced Molecular Detection
Southeast Region Bioinformatics

Creating a Factor in R

Input:

```
 98  # create a factor in r
 99  music_genre <- factor(c("rock", "pop", "pop", "classic", "rock", "jazz"))
100  print(music_genre)
```

Output:

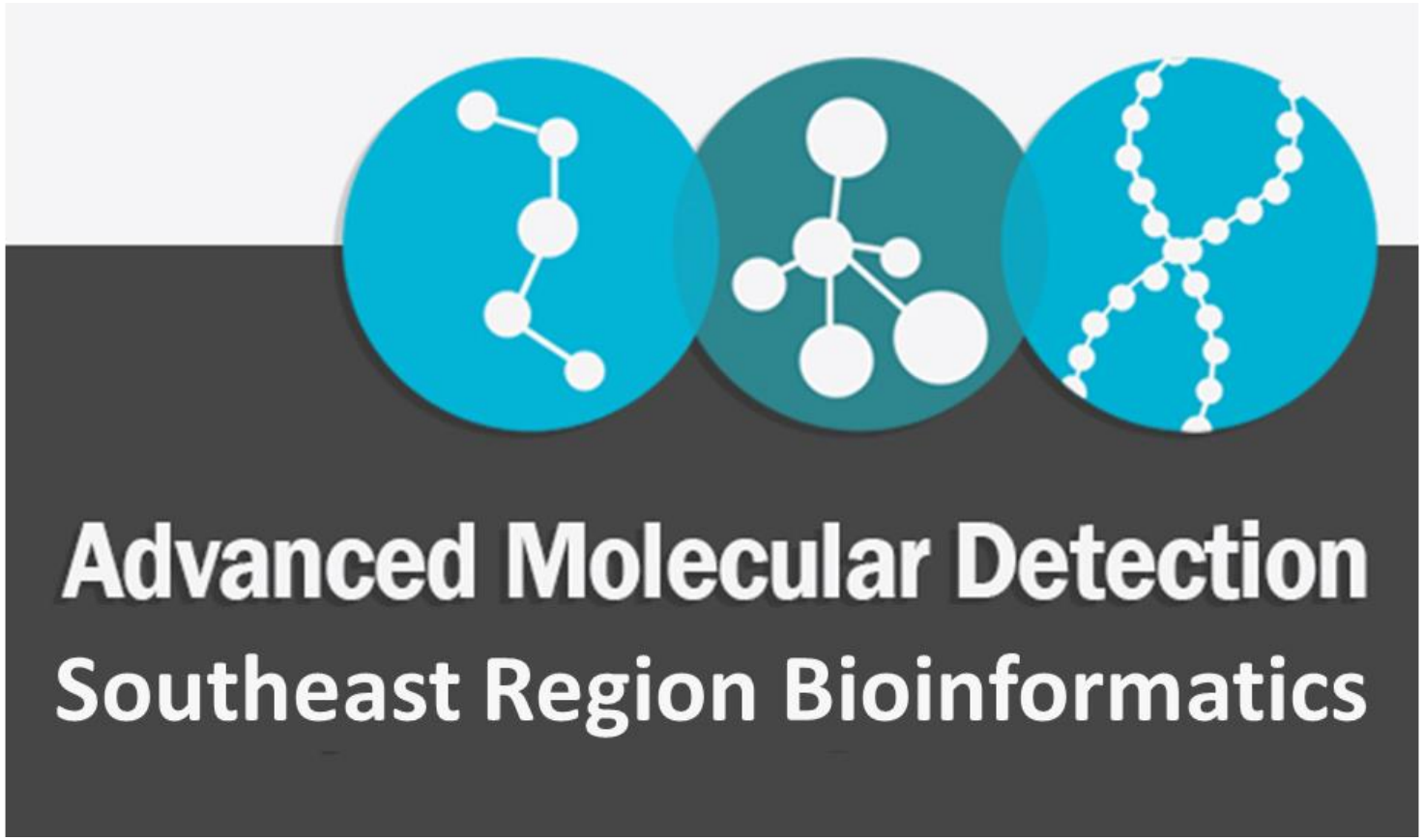
```
> # create a factor in r
> music_genre <- factor(c("rock", "pop", "pop",."classic", "rock", "jazz"))
> print(music_genre)
[1] rock    pop     pop     classic rock    jazz
Levels: classic jazz pop rock
```

Advanced Molecular Detection
Southeast Region Bioinformatics

References to Learn R

- Welcome | R for Data Science (had.co.nz)
- R-intro.pdf (r-project.org)
- R for applied epidemiology and public health | The Epidemiologist R Handbook (epirhandbook.com)

Advanced Molecular Detection
Southeast Region Bioinformatics



Questions?

bphl-sebioinformatics@flhealth.gov

Molly Mitchell, PhD
Bioinformatics Supervisor
Molly.Mitchell@flhealth.gov

Sam Marcellus, MPH
Bioinformatician
Samantha.Marcellus@flhealth.gov

Lakshmi Thsaliki, MS
Bioinformatician
Lakshmi.Thsaliki@flhealth.gov