Week-3: Code-along

Insert your name here

2023-08-30

I. Code to edit and execute

To be submitted on canvas before attending the tutorial

Loading packages

```
# Load package tidyverse
library(tidyverse)
```

Assigning values to variables

```
# Example a.: execute this example
x <- 'A'
x
```

```
# Complete the code for Example b and execute it
x <- 'B'
```

```
# Complete the code for Example c and execute it
x <- 'C'
```

```
# Complete the code for Example d and execute it
x <- 'D'
```

```
# Complete the code for Example e and execute it
x <- 'E'
```

```
# Complete the code for Example f and execute it
X <- 'F'
```

Checking the type of variables

```
# Example a.: execute this example
x <- 'A'
typeof(x)
```

```
# Complete the code for Example b and execute it
x <- 'B'
typeof(x)
```

```
# Complete the code for Example c and execute it
x <- 'C'
typeof(x)
```

```
# Complete the code for Example d and execute it
x <- 'D'
typeof(x)
```

```
# Complete the code for Example e and execute it
x <- 'E'
typeof(x)
```

```
# Complete the code for Example f and execute it
x <- 'F'
typeof(x)
```

Need for data types

```
# import the cat-lovers data from the csv file you downloaded from canvas
cat_lovers <- read.csv('cat-lovers.csv')
```

```
# Compute the mean of the number of cats: execute this command
mean(cat_lovers$number_of_cats)
```

```
# Get more information about the mean() command using ? operator
?mean()
```

```
# Convert the variable number_of_cats using as.integer()
as.integer('number_of_cats')
```

```
# Display the elements of the column number_of_cats
cat_lovers$number_of_cats
```

```
# Display the elements of the column number_of_cats after converting it using as.numeric()
as.numeric(cat_lovers$number_of_cats)
```

Create an empty vector

```
# Empty vector
x <- vector()
# Type of the empty vector
typeof(x)
```

Create vectors of type logical

```
# Method 1
x<-vector("logical",length=5)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
# Method 2
x<-logical(5)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
# Method 3
x<-c(TRUE,FALSE,TRUE,FALSE,TRUE)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

Create vectors of type character

```
x<-vector("character",length=5)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
x<-character(5)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
x<-c('TRUE','FALSE','TRUE','FALSE','TRUE')
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

Create vectors of type integer

```
# Method 1
x<-vector("integer",length=5)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
# Method 2
x<-integer(5)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
x<-c(1L,2L,3L,4L,5L)
# Display the contents of x
print(x)
# Display the type of x
print(typeof(x))
```

```
# Method 4
x<-seq(from=1,to=5,by=1)
# Display the contents of x

# Display the type of x
typeof(x)
```

```
# Method 5
x <- 1:5
# Display the contents of x
typeof(x)
# Display the type of x
```

Create vectors of type double

```
# Method 1

x <- numeric(0)
typeof (x)
```

```
# Method 2

x <- c(12)
typeof(x)
```

```
# Method 3

x <- vector("numeric", length = 0)
typeof(x)
```

Implicit coercion

Example 1

```
# Create a vector
x <- numeric(0)
typeof(x)
```

```
x <- append(x, "pineapple")
typeof(x)
```

Example 2

```
x <- numeric(0)
typeof(x)
```

```
x <- append(x, 6)
typeof(x)
```

Example 3

```
x <- numeric(0)
typeof(x)
```

```
x <- append(x, TRUE)
typeof(x)
```

Example 4

```
x <- numeric(0)
typeof(x)
```

```
x <- append(x, '3')
typeof(x)
```

Explicit coercion

Example 1

```
x <- numeric(0)
typeof(x)
```

```
x <- as.character(x)
typeof(x)
```

Example 2

```
x <- '1'
typeof(x)
```

```
x <- as.double(x)
typeof(x)
```

Accessing elements of the vector

```
# Create a vector
x <- c(1,10,9,8,1,3,5)
```

```
# Access one element with index 3
x3 <- x[3]
```

```
# Access elements with consecutive indices, 2 to 4: 2,3,4
x24<- x[2:4]
```

```
# Access elements with non-consecutive indices, 1,3,5
x135<- c(1, 3, 5)
```

```
# Access elements using logical vector
x[c(TRUE,FALSE,FALSE,TRUE,FALSE,FALSE,TRUE)]
```

```
# Access elements using the conditional operator <
xcondi<- x[x < 5]
```

Examining vectors

```
# Display the length of the vector
print(length(x))
# Display the type of the vector
print(typeof(x))
# Display the structure of the vector
print(str(x))
```

Lists

```
# Initialise a named list
my_pie = list(type="key lime", diameter=7, is.vegetarian=TRUE)
# display the list
my_pie
```

```
# Print the names of the list
element_names <- names(my_pie)
```

```
# Retrieve the element named type
type_element <- my_pie[element_names == "vegetarian"]
```

```
# Retrieve a truncated list
subset_names <- c("type", "flavor", "size")
truncated_pie <- my_pie[subset_names]
```

```
# Retrieve the element named type
type_element <- my_pie$type
print(type_element)
```

Exploring data-sets

```
# Install package
install.packages("openintro")
# Load the package
library(openintro)
# Load package
library(tidyverse)
```

```
# Catch a glimpse of the data-set: see how the rows are stacked one below another
glimpse(loans_full_schema)
```

```
# Selecting numeric variables
loans <- loans_full_schema %>% # <-- pipe operator
  select(paid_total, term, interest_rate,
         annual_income,paid_late_fees,debt_to_income)
# View the columns stacked one below another
glimpse(loans)
```

```
# Selecting categoric variables
loans <- loans_full_schema %>%
  select( ) # type the chosen columns as in the lecture slide
# View the columns stacked one below another
glimpse(loans)
```