Week-3: Code-along

Insert your name here

2023-08-29

I. Code to edit and execute

To be submitted on canvas before attending the tutorial

Loading packages

```
# Load package tidyverse
```

Assigning values to variables

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

```
## [1] "A"
```

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

```
## [1] "Apple"
```

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

```
## [1] FALSE
```

```
# Complete the code for Example d and execute it
x <- 5L
x
```

```
## [1] 5
```

```
# Complete the code for Example e and execute it
x <- 5
x
```

```
## [1] 5
```

```
# Complete the code for Example f and execute it
x <- li
x
```

Checking the type of variables

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

```
## [1] "character"
```

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

```
## [1] "character"
```

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

```
## [1] "logical"
```

```
# Complete the code for Example d and execute it
x <- 5L
typeof(x)
```

```
## [1] "integer"
```

```
# Complete the code for Example e and execute it
x <- 5
typeof(x)
```

```
## [1] "double"
```

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

Need for data types

```
# import the cat-lovers data from the csv file you downloaded from canvas
```

```
# 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(cat_lovers$number_of_cats)
```

```
# Convert the variable number_of_cats using as.integer()
mean(as.integer(cat_lovers$number_of_cats))
```

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

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

Create an empty vector

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

```
## [1] "logical"
```

Create vectors of type logical

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

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

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

```
## [1] "logical"
```

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

```
## [1] FALSE FALSE FALSE FALSE FALSE
```

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

```
## [1] "logical"
```

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

```
## [1]  TRUE FALSE  TRUE FALSE  TRUE
```

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

```
## [1] "logical"
```

Create vectors of type character

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

```
## [1] "" "" "" "" ""
```

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

```
## [1] "character"
```

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

```
# Method 3
x<-c('A','b','r','q')
# 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)
```

```
## [1] 0 0 0 0 0
```

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

```
## [1] "integer"
```

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

```
## [1] 0 0 0 0 0
```

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

```
## [1] "integer"
```

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

```
## [1] 1 2 3 4 5
```

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

```
## [1] "integer"
```

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

```
## [1] 1 2 3 4 5
```

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

```
## [1] "double"
```

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

```
## [1] 1 2 3 4 5
```

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

```
## [1] "integer"
```

Create vectors of type double

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

```
## [1] 0 0 0 0 0
```

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

```
## [1] "double"
```

```
# Method 2
x<-double(5)
# Display the contents of x
print(x)
```

```
## [1] 0 0 0 0 0
```

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

```
## [1] "double"
```

```
# Method 3
x<-c(1.787,0.63573,2.3890)
# Display the contents of x
print(x)
```

```
## [1] 1.78700 0.63573 2.38900
```

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

```
## [1] "double"
```

Implicit coercion

Example 1

```
# Create a vector
x<-c(1.8)
# Check the type of x
typeof(x)
```

```
## [1] "double"
```

```
# Add a character to the vector
x<-c(x,'a')
# Check the type of x
typeof(x)
```

```
## [1] "character"
```

Example 2

```
# Create a vector
x<-c(TRUE)
# Check the type of x
typeof(x)
```

```
## [1] "logical"
```

```
# Add a number to the vector
x<-c(x,2)
# Check the type of x
typeof(x)
```

```
## [1] "double"
```

Example 3

```
# Create a vector
x<-c('a')
# Check the type of x
typeof(x)
```

```
## [1] "character"
```

```
# Add a logical value to the vector
x<-c(x,TRUE)
# Check the type of x
typeof(x)
```

```
## [1] "character"
```

Example 4

```
# Create a vector
x<-c(1L)
# Check the type of x
typeof(x)
```

```
## [1] "integer"
```

```
# Add a number to the vector
x<-c(x,2)
# Check the type of x
typeof(x)
```

```
## [1] "double"
```

Explicit coercion

Example 1

```
# Create a vector
x<-c(1L)
# Check the type of x
typeof(x)
```

```
## [1] "integer"
```

```
# Convert the vector to type character
x<-as.character(x)
# Check the type of x
typeof(x)
```

```
## [1] "character"
```

Example 2

```
# Create a vector
x<-c('A')
# Check the type of x
typeof(x)
```

```
## [1] "character"
```

```
# Convert the vector to type double
x<-as.numeric(x)
```

```
## Warning: NAs introduced by coercion
```

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

```
## [1] "double"
```

Accessing elements of the vector

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

```
# Access one element with index 3
x[3]
```

```
## [1] 9
```

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

```
## [1] 10  9  8
```

```
# Access elements with non-consecutive indices, 1,3,5
x[c(1,3,5)]
```

```
## [1] 1 9 1
```

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

```
## [1] 1 8 5
```

```
# Access elements using the conditional operator <
x[x<10]
```

```
## [1] 1 9 8 1 3 5
```

Examining vectors

```
# Display the length of the vector
print(length(x))
```

```
## [1] 7
```

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

```
## [1] "double"
```

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

```
##  num [1:7] 1 10 9 8 1 3 5
## NULL
```

Lists

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

```
## $type
## [1] "key lime"
##
## $diameter
## [1] 7
##
## $is.vegetarian
## [1] TRUE
```

```
# Print the names of the list
names(my_pie)
```

```
## [1] "type"          "diameter"      "is.vegetarian"
```

```
# Retrieve the element named type
my_pie$type
```

```
## [1] "key lime"
```

```
# Retrieve a truncated list
my_pie["type"]
```

```
## $type
## [1] "key lime"
```

```
# Retrieve the element named type
my_pie[["type"]]
```

```
## [1] "key lime"
```

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(grade,state,homeownership,disbursement_method) # type the chosen columns as in the lect
ure slide
# View the columns stacked one below another
glimpse(loans)
```