

# Week-3: Code-along

Esther Kho Yining

2024-01-30

## 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'
```

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

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

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

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

```
# Complete the code for Example f and execute it  
x <- 1i
```

### 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 <- 1i
typeof(x)
```

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

Need for data types

```
# import the cat-lovers data from the csv file you downloaded from canvas
library(tidyverse)
read_csv("cat_lovers.csv")
```

```
## # A tibble: 60 x 3
##   name          number_of_cats handedness
##   <chr>         <chr>         <chr>
## 1 Bernice Warren 0          left
## 2 Woodrow Stone 0          left
## 3 Willie Bass   1          left
## 4 Tyrone Estrada 3          left
## 5 Alex Daniels  3          left
## 6 Jane Bates    2          left
## 7 Latoya Simpson 1          left
## 8 Darin Woods   1          left
## 9 Agnes Cobb    0          left
## 10 Tabitha Grant 0          left
## # i 50 more rows
```

```
# Assign the variable to the dataset
cat_lovers <- read_csv("cat_lovers.csv")
```

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

```
## Warning in mean.default(cat_lovers$number_of_cats): argument is not numeric or
## logical: returning NA
```

```
## [1] NA
```

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

```
knitr::include_graphics("/Users/EstherKho/Documents/Y2S2/NM2207/Week 3/Code-along-3/mean.jpg")
```

The screenshot shows an RStudio interface with a code editor on the left, a console at the bottom, and a sidebar on the right displaying the R documentation for the `mean` function.

**Code Editor:** The code chunk contains the following R code:

```
106 # Assign the variable to the dataset
107 cat_lovers <- read_csv("cat_lovers.csv")
108 ~~~
109
110 ~~~{r,warning=TRUE,message=FALSE,eval=FALSE,echo=TRUE}
111 # Compute the mean of the number of cats: execute this command
112 mean(cat_lovers$number_of_cats)
113 ~~~
```

**Console:** The console shows the execution of the code, including a warning message:

```
Warning: argument is not numeric or logical: returning NA[1] NA
```

**Environment:** The environment pane shows the `cat_lovers` data frame with 60 observations and 3 variables: `name`, `number_of_cats`, and `handedness`.

**R Documentation:** The sidebar displays the R documentation for the `mean` function, titled "Arithmetic Mean". It includes a description, usage, arguments, and value sections.

**Usage:**

```
mean(x, ...)
```

**Arguments:**

- `x`: An R object. Currently there are methods for numeric/logical vectors and [date](#), [date-time](#) and [time interval](#) objects. Complex vectors are allowed for `trim = 0`, only.
- `trim`: the fraction (0 to 0.5) of observations to be trimmed from each end of `x` before the mean is computed. Values of trim outside that range are taken as the nearest endpoint.
- `na.rm`: a logical evaluating to `TRUE` or `FALSE` indicating whether NA values should be stripped before the computation proceeds.
- `...`: further arguments passed to or from other methods.

**Value:**

If `trim` is zero (the default), the arithmetic mean of the values in `x` is computed, as a numeric or complex vector of length one. If `x` is not logical (coerced to numeric), numeric (including integer) or complex, `NA_real_` is returned, with a warning.

If `trim` is non-zero, a symmetrically trimmed mean is computed with a fraction of `trim` observations

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

```
## Warning in mean(as.integer(cat_lovers$number_of_cats)): NAs introduced by
## coercion
```

```
## [1] NA
```

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

```
## [1] "0"  
## [2] "0"  
## [3] "1"  
## [4] "3"  
## [5] "3"  
## [6] "2"  
## [7] "1"  
## [8] "1"  
## [9] "0"  
## [10] "0"  
## [11] "0"  
## [12] "0"  
## [13] "1"  
## [14] "3"  
## [15] "3"  
## [16] "2"  
## [17] "1"  
## [18] "1"  
## [19] "0"  
## [20] "0"  
## [21] "1"  
## [22] "1"  
## [23] "0"  
## [24] "0"  
## [25] "4"  
## [26] "0"  
## [27] "0"  
## [28] "0"  
## [29] "0"  
## [30] "0"  
## [31] "0"  
## [32] "0"  
## [33] "0"  
## [34] "0"  
## [35] "0"  
## [36] "0"  
## [37] "0"  
## [38] "0"  
## [39] "0"  
## [40] "0"  
## [41] "0"  
## [42] "0"  
## [43] "1"  
## [44] "3"  
## [45] "3"  
## [46] "2"  
## [47] "1"  
## [48] "1.5 - honestly I think one of my cats is half human"  
## [49] "0"
```

```
## [50] "0"
## [51] "1"
## [52] "0"
## [53] "1"
## [54] "three"
## [55] "1"
## [56] "1"
## [57] "1"
## [58] "0"
## [59] "0"
## [60] "2"
```

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

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

```
## [1] 0 0 1 3 3 2 1 1 0 0 0 0 1 3 3 2 1 1 0 0 1 1 0 0 4
## [26] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 3 3 2 1 NA 0 0
## [51] 1 0 1 NA 1 1 1 0 0 2
```

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)
```

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

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

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

```
# Method 3
x<-c('A','b','r','q')
# Display the contents of x
print(x)
```

```
## [1] "A" "b" "r" "q"
```

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

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

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(1,2,3,4,5)
# Display the contents of x
print(x)
```

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

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

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

```
# Method 4  
x<-seq(from=1,to=5,by=0.1)  
# Display the contents of x  
print(x)
```

```
## [1] 1.0 1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2.0 2.1 2.2 2.3 2.4 2.5 2.6 2.7 2.8  
## [20] 2.9 3.0 3.1 3.2 3.3 3.4 3.5 3.6 3.7 3.8 3.9 4.0 4.1 4.2 4.3 4.4 4.5 4.6 4.7  
## [39] 4.8 4.9 5.0
```

```
# 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

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

### Example 1

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

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

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

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

### Example 2

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

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

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

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

### Example 3

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

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

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

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

### Example 4

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

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

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

## Explicit coercion

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

### Example 1

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

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

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

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

### Example 2

```
## [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
```

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

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

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

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

```
# 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)
```

Exploring data-sets