R Bootcamp

August 23-24, 2021

UBC MFRE | Master of Food and Resource Economics

Learning Objectives

- To understand the usefulness of functions in programming
- To define a function
- To set default values in the arguments of functions
- Understand how to use UN ComTrade user written function

UBC
MFRE

Resources

- Creating Functions in Programming with R, Software Carpentry
- Functions, R for Data Science

UBC MFRE

Functions

- Why do we use functions?
  - Reduce likelihood of errors!
  - Repeat operations with just a single command instead of copy and pasting
  - Perform tasks more efficiently

UBC MFRE

Functions

- A function is a set of code that performs a particular task when called.[1]

UBC MFRE

[1] https://www.w3schools.com/r/r_functions.asp

Why use functions?

- What does this code chunk do?
- Do you notice any error?

```
df$var1_c <- (df$var1_f – 32) * 5 / 9
df$var2_c <- (df$var1_f – 32) * 5 / 9
df$var3_c <- (df$var3_f – 32) * 5 / 9
df$var4_c <- (df$var4_f – 32) * 5 / 9
```

UBC MFRE

https://r4ds.had.co.nz/functions.html

Why use functions?

- The code chunk converts 4 variables from Fahrenheit to celcius.
- But in copying and pasting the codes, I forgot to change var1_f to var2_f
- This mistake is **very common**

```
df$var1_c <- (df$var1_f – 32) * 5 / 9
df$var2_c <- (df$var1_f – 32) * 5 / 9
df$var3_c <- (df$var3_f – 32) * 5 / 9
df$var4_c <- (df$var4_f – 32) * 5 / 9
```

UBC MFRE

Functions

- Instead of copying and pasting, we can write a function that we can use multiple types
- We will define a function called f_to_c that converts temperatures from Fahrenheit to Celcius

```
f_to_c <- function(temp_f){
  temp_c <- (temp_f - 32) * 5 / 9
  return(temp_c)
}
```

UBC MFRE

Functions

- We begin a function by typing

function_name <- function(argument_names)

- In our example, we call our function **f_to_c** that takes the Fahrenheit temperature (temp_f) as an arumgent
  - **f_to_c <- function(temp_f)**

UBC MFRE

Functions

- Next, we add in curly brackets ({}) that will contain the body of the function, which are the codes that will be executed when we call the function.
  - **f_to_c <- function(temp_f){**

    code_chunk

    }

UBC MFRE

Functions

- Then we add the code statments. We typically indent the statements for readability.
  - **f_to_c <- function(temp_f){**
    temp_c <- (temp_f – 32) * 5 / 9
    return(temp_c)
    }
- This function creates a temporary variable called temp_c that takes the parameter provided (temp_f) and performs the calculation. Then it returns temp_c.

UBC MFRE

Functions

- Give it a try!

```
# convert 32F to C
f_to_c(32)
```

```
> f_to_c(32)
[1] 0
```

```
# convert 150F to C
f_to_c(150)
```

```
> f_to_c(150)
[1] 65.55556
```

UBC MFRE

Give it a try!

- Create a new function called *square* that takes 2 values (var1 and var2). Var1 will be the base, and var2 will be the power.
- Then test your function: if var1 = 2 and var2 = 3, the result should be 8

UBC MFRE

Give it a try!

```
square <- function(var1, var2){
  result <- var1^var2
  print(result)
}

square(2,3)
```

UBC MFRE

Setting default values

- We can also set default values.
- Let's try to use the **square()** function we just created. But now let's just provide one argument, instead of 2. What happens?

```
square(2)
```

UBC MFRE

Setting default values

- We can explicitly assign values to the function, such as var2 = 3. This means that if no value is provided for var2, then it will use 3 as default. If a value for var2 is provided, then the function will use that value.

- Let's edit the **square()** function to add a default value.

UBC MFRE

Setting default values

```
square <- function(var1, var2 = 3){
  result <- var1^var2
  print(result)
}

square(2)
square(2, 4)
```

UBC MFRE

It takes practice

- It will take time to get used to writing functions.
- At times, it may feel that writing functions may take a longer time than copying and pasting. However, the benefits are worth it!

UBC
MFRE

UN Comtrade function

- Sometimes, you may be asked to use a function that has already been coded for you. One example is get.Comtrade() function. Some documentation here
- This function extracts trade data from the UN Comtrade database.
- You will not need to learn how to code such a function that is as complicated as the get.Comtrade() function, but you will need to learn how to read the documentation to use it.

UBC MFRE

A user-defined function to extract data from the UN Comtrade API

The function defined in this example, `get.Comtrade()`, extracts data from UN Comtrade using either the `csv` or the `json` format.

```
get.Comtrade <- function(url="http://comtrade.un.org/api/get?"
                         ,maxrec=50000
                         ,type="C"
                         ,freq="A"
                         ,px="HS"
                         ,ps="now"
                         ,r
                         ,p
                         ,rg="all"
                         ,cc="TOTAL"
                         ,fmt="json"
)
{
  string<- paste(url
                 ,"max=",maxrec,"&" #maximum no. of records returned
                 ,"type=",type,"&" #type of trade (c=commodities)
                 ,"freq=",freq,"&" #frequency
                 ,"px=",px,"&" #classification
                 ,"ps=",ps,"&" #time period
                 ,"r=",r,"&" #reporting area
                 ,"p=",p,"&" #partner country
                 ,"rg=",rg,"&" #trade flow
                 ,"cc=",cc,"&" #classification code
                 ,"fmt=",fmt        #Format
                 ,sep = ""
  )

  if(fmt == "csv") {
    raw.data<- read.csv(string,header=TRUE)
    return(list(validation=NULL, data=raw.data))
  } else {
    if(fmt == "json" ) {
      raw.data<- fromJSON(file=string)
      data<- raw.data$dataset
      validation<- unlist(raw.data$validation, recursive=TRUE)
      ndata<- NULL
      if(length(data)> 0) {
        var.names<- names(data[[1]])
        data<- as.data.frame(t( sapply(data,rbind)))
        ndata<- NULL
        for(i in 1:ncol(data)){
          data[sapply(data[,i],is.null),i]<- NA
          ndata<- cbind(ndata, unlist(data[,i]))
        }
        ndata<- as.data.frame(ndata)
        colnames(ndata)<- var.names
      }
      return(list(validation=validation,data =ndata))
    }
  }
}
```


```
get.Comtrade <- function(url="http://comtrade.un.org/api/get?"
                         ,maxrec=50000
                         ,type="C"
                         ,freq="A"
                         ,px="HS"
                         ,ps="now"
                         ,r
                         ,p
                         ,rg="all"
                         ,cc="TOTAL"
                         ,fmt="json"
)
```


UBC MFRE

UN Comtrade function

```
get.Comtrade <- function(url="http://comtrade.un.org/api/get?"
                         ,maxrec=50000
                         ,type="C"
                         ,freq="A"
                         ,px="HS"
                         ,ps="now"
                         ,r
                         ,p
                         ,rg="all"
                         ,cc="TOTAL"
                         ,fmt="json"
)
```

- Maxrec = max records
- Type = C is for commodity (currently the only one available)
- Freq = A is for annual, M is for monthly
- Px = “HS” is the classification
- Ps = “now”
- R = region
- P = partner
- Rg = “all” is the default, 1 = imports, 2 = exports
- cc = classification code
- Fmt = json or csv format

UBC MFRE

UN Comtrade Function

- Let's try to download the following data:
  - Netherlands (r = "528") as the reporting country
  - Netherlands/reporting country (0), Indonesia (170), Malaysia (320), Columbia (360), Guatemala (458)
  - ps = year (e.g., ps = "2020" for the year 2020)
  - rg = 1 for imports
  - cc = "151110" for crude palm oil (see here for others)

```
raw <- get.Comtrade(r="528", p="0,170,320,360,458", ps="2020",
                    rg=1, cc="151110")
```

UBC MFRE

Recap

- Showed how functions can help reduce coding errors
- Defined a function that converted Fahrenheit to Celcius
- Set default values in functions
- Applied knowledge on functions with the get.Comtrade() function

UBC MFRE

UBC MFRE

mfre.landfood.ubc.ca