390R R programming
11/17/2020
Notes

Lets say you have a basic plot that you want to see with different colors.

```
ggplot(diamonds)+geom_bar(aes(x=cut),fill=”black”)
plot_cut<- function(fill_cut= “red”){
p<-ggplot(diamonds)+ geom_bar(aes(x=cut),fill=fill_cut) return(p)}
```

Now you can just enter plot_cut(“blue”) and with other colors to produce the chart with different colors.

Question: Make a function: it takes a single number as input and returns TRUE if the number is divisible 9 and FALSE if not.

```
 Div.by.9 <- function(x){ return( x%%9 == 0) }
```

Do.call is quite useful if you want to specify the name of a function as a character.
You use do.call to specify the name of a function.

```
do.call(min_from_midnight, args = list(time_hhmm = 1030))
```
this also calls the function

You can have a function that does different things depending on your input.

```
Run_this <- function(x, func = mean){ do.call(func,args = list(x)) }
```

Default function finds the mean, but you can call run_this(1:10,sum) to get the sum

Simple if else statement

```
x<- 1
if(x==1){
        print(“Hello”)
} else{
print(“Goodbye”)
}
```

Suppose you want to display the distribution of cut variable in diamonds data.

```
ggplot(diamonds) + geom_bar(aes(x=cut)) + facet_wrap(~color)
```
OR
```
ggplot(diamonds) + geom_bar(aes(x=cut, fill=color),position=”dodge”)
```

You can combine these into a function based on facet

```
Plot_cut_color <- function(facet = TRUE){
if(facet){ ggplot(diamonds) + geom_bar(aes(x=cut)) + facet_wrap(~color)
} else {
ggplot(diamonds) + geom_bar(aes(x=cut, fill=color),position=”dodge”)
}}
```

Else if statements
You can use else if for multiple conditions

```
Check_hello <- function(x){
if(x==1){
        print(“Hello”)
} else if (x==0){
        print(“confused”)
} else {
print(“Goodbye”)
}
```

Question: Build a grading function that takes the score as input, output A if score>=85, B if 85>score>=75, and C if score < 75.

```
Grade <- function(score){
        if(score >=85){
                Grade <- “a”
        }else if (score >=75){
                Grade <-”b”
        } else {
                Grade <- “c” }
return(grade)
}
```

When you have many options to choose from it’s less efficient to use if else statements. Another statement you can use is the switch function

```
use_switch<- function(x){
switch(x, “a” = “first”, “b” = “second”, “z” = “last”, “c” = “third”, “other”)}
```

This function checks if x is equal to any of the options. If it’s not it will default to the last one.

ifelse() that takes 3 argument and returns a value
ifelse(logical operation, value if operation is true, value if operation is false)
ifelse(x==1, “Yes”,”No”)
Ifelse functions can apply to entire vectors at once.

A more general version of ifelse is case_when()
This is like an extension of ifelse when there are more than 2 options.

```
case_when(
x%% 35 ==0 ~ “fizzbuzz”,
x%%5 == 0 ~ “fizz”,
x%%7 ==0 ~ “buzz”,
TRUE ~ as.character(x)
)
```
This goes through each line to check the first one that works and outputs that value.

Question: Make the grading function in the previous exercise to work for a vector score c(70,90,80,85,95)
```
grade<- function(x){
case_when(x>= 85 ~ “A”,
          x>= 75 ~ “B”,
          TRUE ~ “C”)
}
```

Loops
I don’t recommend using loops in R because they are inefficient. Sometimes you will have to use for loops.
```
for(i in 1:100){
print(i)
}
```
ALSO
```
Fruit <- c(“apple”,”banana”,”grape”)
for(i in fruit){
          print(str_length(i))
}
```