2025_4_3_CodingChallengeSix_mer0127

Madeline Redd

2025 - 04 - 06

Question One

Regarding reproducibility, what is the main point of writing your own functions and iterations?

Answer: Creating functions allows others opportunity to see each step of the calculation and will reduce human error if the operations had to be constantly retyped. It could benefit someone when reviewing the code and if needed to build on the code. For iterations, it allows replications to be accurate and efficiently if preformed exactly the same, but also allows other to test different parameters.

Question Two

In your own words, describe how to write a function and a for loop in R and how they work. Give me specifics like syntax, where to write code, and how the results are returned.

Answer:

Example of Creating a Function Code –>

```
new_function <- function(argument_one, argument_two) { answer <- argument_one
* (argument_two + 87) #Operations to perform with arguments to produce results
return(answer)
}
```

```
new_function(45, 3)
```

Example of FOR LOOP Code –>

```
iteration_length <- lage-fage+1 # Define the iteration length
```

```
#Necessary Parameters for the For Loop #Best for stable coefficients or numbers that might
need to be adjusted and help reduce human error
```

```
M=0.4 #Natural Mortality survship0=numeric(iteration_length) #Empty numeric vector
for data survship0[1]=1 #survivorship at age 1
```

```
#a is the loop variable #the 2:iteration_length: the sequence it will iterate over
```

for (a in 2:iteration_length){ survship0[a]=survship0[a-1]*exp(-M) #Writing calculation code could be built into almost anything.

}

print(survship0) #print data

```
#Full code for Mortality with Fishing Pressure on Atlantic Tarpon
#Parameters for for loop
fage=1 #first age
lage=8 #last age
linf=360 #Maximum length
k=0.45 # growth coefficient
t0=0 #age 0
cv_tl=0.1 #variation in total length
M=0.4 #Natural mortality
l50_mat=225 #length at 50% pop of maturity
h_mat=0.05 #steepness of maturity
a_wt=2e-5 #length-weight
b_wt=3 #length-weight
alpha_hat=8 #recruitment
R0=1 #initial recruitment
tmax=100 #max years
F_full=0.5 #Fishing Mortality
MLL=250 #Minimum legal length

age=fage:lage
n_ages=lage-fage+1
length=linf*(1-exp(-k*(age-t0)))
weight=a_wt*length^b_wt
maturity=1/(1+exp(-h_mat*(length-l50_mat)))
fecundity=weight*maturity
sd_length=cv_tl*length
vuln=1-pnorm(MLL,length,sd_length)

survship0=numeric(n_ages)
survship0[1]=1

#Survivorship For Loop
for (a in 2:n_ages){
  survship0[a]=survship0[a-1]*exp(-M)
}
survship0[lage]=survship0[lage]/(1-exp(-M))

ssb0=sum(survship0*fecundity)
alpha=alpha_hat/ssb0
```

```
beta=log(alpha*ssb0)/(R0*ssb0)


N_a=matrix(NA,nrow=tmax,ncol=n_ages) #Blank matrix for values to be imported into
F_a=N_a
Z_a=N_a
U_a=N_a
yield_t=numeric(tmax)
ssb_t=yield_t
SPR_t=yield_t

N_a[1,]=R0*survship0
F_a[1,]=F_full*vuln
Z_a[1,]=F_a[1,]+M
U_a[1,]=1-exp(-F_a[1,])
yield_t[1]=sum(N_a[1,]*weight*U_a[1,])
ssb_t[1]=sum(N_a[1,]*fecundity)
SPR_t[1]=ssb_t[1]/(R0*ssb0)

for(t in 2:tmax){
  N_a[t,1]=alpha*ssb_t[t-1]*exp(-beta*ssb_t[t-1])
  for(a in 2:n_ages){
    N_a[t,a]=N_a[t-1,a-1]*exp(-Z_a[t-1,a-1])
  }
  N_a[t,n_ages]=N_a[t,n_ages]+N_a[t-1,a]*exp(-Z_a[t-1,a])
  F_a[t,]=F_full*vuln
  Z_a[t,]=F_a[t,]+M
  U_a[t,]=1-exp(-F_a[t,])
  yield_t[t]=sum(N_a[t,]*weight*U_a[t,])
  ssb_t[t]=sum(N_a[t,]*fecundity)
  SPR_t[t]=ssb_t[t]/(R0*ssb0)
}

windows()
plot(SPR_t,type='l',ylim=c(0,1))
```

Question Three

Read in the Cities.csv file from Canvas using a relative file path.

```
CITIES <- read.csv("Cities.csv")
head(CITIES)
```

```
##           city   city_ascii state_id state_name county_fips county_name     lat
## 1    New York    New York        NY   New York       36081       Queens 40.6943
## 2 Los Angeles Los Angeles        CA California        6037 Los Angeles 34.1141
## 3     Chicago     Chicago        IL   Illinois       17031        Cook 41.8375
## 4       Miami       Miami        FL    Florida       12086  Miami-Dade 25.7840
## 5     Houston     Houston        TX      Texas       48201      Harris 29.7860
## 6      Dallas      Dallas        TX      Texas       48113      Dallas 32.7935
##        long population density
## 1  -73.9249   18832416 10943.7
## 2 -118.4068   11885717  3165.8
## 3  -87.6866    8489066  4590.3
## 4  -80.2101    6113982  4791.1
## 5  -95.3885    6046392  1386.5
## 6  -96.7667    5843632  1477.2
```

Question Four

Write a function to calculate the distance between two pairs of coordinates based on the Haversine formula (see below). The input into the function should be lat1, lon1, lat2, and lon2. The function should return the object distance_km. All the code below needs to go into the function.

#GIVEN CODE –>

#Convert to radians rad.lat1 <- lat1 * pi/180 rad.lon1 <- lon1 * pi/180 rad.lat2 <- lat2 * pi/180 rad.lon2 <- lon2 * pi/180

#Haversine Formula delta_lat <- rad.lat2 - rad.lat1 delta_lon <- rad.lon2 - rad.lon1 a <- sin(delta_lat / 2)^2 + cos(rad.lat1) * cos(rad.lat2) * sin(delta_lon / 2)^2 c <- 2 * asin(sqrt(a))

earth_radius <- 6378137 #Earth's radius in kilometers

distance_km <- (earth_radius * c)/1000 #Calculate the distance

```r
#Function Code

Calculate_Distance <-function (lat1, lon1, lat2, lon2){

rad.lat1 <- lat1 * pi/180
rad.lon1 <- lon1 * pi/180
rad.lat2 <- lat2 * pi/180
rad.lon2 <- lon2 * pi/180

delta_lat <- rad.lat2 - rad.lat1
delta_lon <- rad.lon2 - rad.lon1
a <- sin(delta_lat / 2)^2 + cos(rad.lat1) * cos(rad.lat2) * sin(delta_lon / 2)^2
c <- 2 * asin(sqrt(a))

Earth_Radius <- 6378137

distance_km <- (Earth_Radius * c)/1000
return (distance_km)
}
```

Question Five

Using your function, compute the distance between Auburn, AL and New York City a. Subset/filter the Cities.csv data to include only the latitude and longitude values you need and input as input to your function. b. The output of your function should be 1367.854 km

```
AU_lat=CITIES$lat[CITIES$city=="Auburn"]
AU_lon=CITIES$long[CITIES$city=="Auburn"]
NY_lat=CITIES$lat[CITIES$city=="New York"]
NY_lon=CITIES$long[CITIES$city=="New York"]

Calculate_Distance(AU_lat,AU_lon, NY_lat, NY_lon)
```

```
## [1] 1367.854
```

Question Six

Now, use your function within a for loop to calculate the distance between all other cities in the data. The output of the first 9 iterations is shown below.

```
# Calculate distances and add to csv file

VALUES<- unique(CITIES$city)

distances <- numeric(length(VALUES))

 for (i in seq_along(VALUES)) {
   lat1 <- CITIES$lat[CITIES$city == "Auburn"]
   lon1 <- CITIES$lon[CITIES$city == "Auburn"]
   lat2 <- CITIES$lat[CITIES$city == VALUES[i]]
   lon2 <- CITIES$lon[CITIES$city == VALUES[i]]
   distances[i] <- Calculate_Distance(lat1, lon1, lat2, lon2)
 }
```

```r
CITIES$distance_from_Auburn <- distances

head(CITIES)
```

```
##          city  city_ascii state_id state_name county_fips county_name     lat
## 1    New York    New York       NY   New York       36081      Queens 40.6943
## 2 Los Angeles Los Angeles       CA California        6037 Los Angeles 34.1141
## 3     Chicago     Chicago       IL   Illinois       17031        Cook 41.8375
## 4       Miami       Miami       FL    Florida       12086  Miami-Dade 25.7840
## 5     Houston     Houston       TX      Texas       48201      Harris 29.7860
## 6      Dallas      Dallas       TX      Texas       48113      Dallas 32.7935
##        long population density distance_from_Auburn
## 1  -73.9249   18832416 10943.7            1367.8540
## 2 -118.4068   11885717  3165.8            3051.8382
## 3  -87.6866    8489066  4590.3            1045.5213
## 4  -80.2101    6113982  4791.1             916.4138
## 5  -95.3885    6046392  1386.5             993.0298
## 6  -96.7667    5843632  1477.2            1056.0217
```

Bonus Points

Bonus point if you can have the output of each iteration append a new row to a dataframe, generating a new column of data. In other words, the loop should create a dataframe with three columns called city1, city2, and distance_km, as shown below. The first six rows of the dataframe are shown below.

Question Seven

Commit and push a gfm .md file to GitHub inside a directory called Coding Challenge 6. Provide me a link to your github written as a clickable link in your .pdf or .docx

Coding Challenge Six