Training

Murtiza

9/18/2020

Difference in Function Framework

Function objective: Giving x and y, find the greatest value among them.

```
# Standard R
function_standard <- function(x, y){
  if (x > y) {
    z <- x
  } else {
    z <- y
  }
  return(z)
}

# Dplyr
function_dplyr <- function(x,y){
  z <- if_else(x > y, x, y)
  return(z)
}
```

Introduction of some features with real examples

Finding the norm of a vector. (L1, L2)

```
norm_dplyr <- function(vector, method = "L2"){  # Add a default method as L2 if there is no input for m

  # Add an error message with the if statement. It is useful when you write a complex function especial
  if(!(method %in% c("L1", "L2"))){
    return("Method must be either 'L1' or 'L2'")
  }

  L1 <- vector %>% abs() %>% sum() # L1 norm
  L2 <- vector^2 %>% sum() %>% sqrt() # L2 norm
  norm_ <- ifelse(method == "L2",
                  L2,
                  L1)
  return(norm_)
}
```

```r
A <- c(3, 4,5,6,7)
norm_dplyr(A)
```

```
## [1] 11.61895
```

```r
norm_dplyr(A, "L2")
```

```
## [1] 11.61895
```

Data processing Examples

```r
# Check the missing values and inappropriate values
employee <- c('John','Peter','Jolie', 'Adam',NA)
salary <- c(21000, 23400, NA, 35000, -33000)
Gender <- c('Male', 'Male', 'Female', NA , 'Female')
emply_data <- data.frame(employee,salary, Gender) %>% mutate_if(is.factor, as.character)

# 1) check the NA values
Check_NA <- function(DF){
  DF %>%
  select_if(function(x) any(is.na(x))) %>% # finding the NA values
  summarise_each(funs(sum(is.na(.)))) # summarize

}
Check_NA(emply_data)
```

```
##   employee salary Gender
## 1        1      1      1
```

```r
#  2) check the index of NA values
Check_NA_index <- function(DF){
  DF %>%
  select_if(function(x) any(is.na(x))) %>% # finding the NA values
  summarise_each(funs(which(is.na(.)))) # summarize

}
Check_NA_index(emply_data)
```

```
##   employee salary Gender
## 1        5      3      4
```

```r
# 3) Check inappropriate values, ex. negative values, texts in numbers
summarize_invalid <- function(DF){
 DF %>% summarise(
   na_nagitive_count = sum(is.na(salary) | salary <=0))
}

summarize_invalid(emply_data)
```

```
##   na_nagitive_count
## 1                 2
```

```r
# 4) check with exact index
check_invalid <- function(DF, num_vector = salary){
 DF %>% group_by(employee) %>%
  summarise(Error= case_when(first(which(is.na({{num_vector}}))) == 1 ~ "Na values",
                             first({{num_vector}} <=0) ~ "Negative or zero values",
                             TRUE ~ "-"))
}

check_invalid(emply_data, salary) %>% kable()
```

employee	Error
Adam	-
John	-
Jolie	Na values
Peter	-
NA	Negative or zero values

Functions with for loop

```r
library(gtrendsR)
library("tseries")
library("forecast")
library("ggplot2")
library(ggpubr)

GT_data_weekly <- function(serach_terms = NA, origin_code,
                           language = "en",
                           category =0){

  # Make a dataframe first.

  # Approach 1 (Destination as keywords, category as 555)
  GT_1<-gtrends(keyword = serach_terms,geo=c(origin_code),
                category = category,
                time="today+5-y",hl=language,
                gprop = "web", onlyInterest = TRUE)[[1]] %>%
    select(date, hits) %>%
    rename(Approach_1_hits = hits)
  # Approach 2 (Destination as keywords, without category)
  GT_2<-gtrends(keyword = serach_terms,geo=c(origin_code),
                time="today+5-y",hl=language,
                gprop = "web", onlyInterest = TRUE)[[1]]%>%
    select(date, hits) %>%
    rename(Approach_2_hits = hits)
  # Aprroach 3 (Category as 555 Without any keywords)
  GT_3<-gtrends(geo=c(origin_code),
                category = category,
                time="today+5-y",hl=language,
                gprop = "web", onlyInterest = TRUE)[[1]]%>%
    select(date, hits) %>%
    rename(Approach_3_hits = hits)
```

```
  GT_data <- GT_1 %>%
    left_join(GT_2, by = "date") %>%
    left_join(GT_3, by = "date")

  return(GT_data)
}


# predict and store as a list
Predicting_GT <- function(GT_data){
  # make it TS
  GT_ts1 <- GT_data %>% ungroup() %>%
    select(Approach_1_hits) %>%
    ts(start = c(2015, 09, 20), frequency = 52)
  # make it TS
  GT_ts2 <- GT_data %>% ungroup() %>%
    select(Approach_2_hits) %>%
    ts(start = c(2015, 09, 20), frequency = 52)
  # make it TS
  GT_ts3 <- GT_data %>% ungroup() %>%
    select(Approach_3_hits) %>%
    ts(start = c(2015, 09, 20), frequency = 52)

  # fit and predictions
  forecast_GT <- list()
  for (tsdata in list(GT_ts1, GT_ts2,GT_ts3)) {
    fit <- auto.arima(log(tsdata+1),
                      trace=F,
                      stepwise = T,
                      stationary=T,
                      approximation=FALSE,
                      seasonal = F)
    name <- paste('prediction of: ',colnames(tsdata),sep='')

    # forecast
    arima_forecast <- forecast::forecast(fit, h = 156, lambda = 0, biasadj = TRUE)
    forecast_GT[[name]] <- arima_forecast



  }
  tsdata <- list(GT_ts1, GT_ts2,GT_ts3)
  return(list(tsdata,forecast_GT))

}

# visualization
Predict_plot <- function(serach_terms , origin_code,
                         language,category){
  # First function
  GT_data <- GT_data_weekly(serach_terms, origin_code,
         language, category)
  # Scond function
```

```r
  B <- Predicting_GT(GT_data)


  # GT 1
  GT_p1 <- autoplot(B[[1]][[1]]) + autolayer(B[[2]][[1]])+
            labs(title ="Approach 1", y = "", x= "") +
            ylim(0,100)
  # GT 2
  GT_p2 <- autoplot(B[[1]][[2]]) + autolayer(B[[2]][[2]])+
            labs(title ="Approach 2", y = "",x= "") +
            ylim(0,100)
  # GT 3
  GT_p3 <- autoplot(B[[1]][[3]]) + autolayer(B[[2]][[3]]) +
            labs(title ="Approach 3", y = "",x= "") +
            ylim(0,100)

  ggarrange(GT_p1,GT_p2,GT_p3,
            heights = c(2, 2),ncol = 1, nrow = 3)

}

# Implementation
Predict_plot(serach_terms = 'Germany', origin_code = 'SY',
         language = "en",
         category = 555)
```