READ ME

Data

I downloaded the "FAO.csv" from Kaggle.com *https://www.kaggle.com/dorbicycle/world-foodfeed-production* on December 10 2017. I placed it in the **Data** directory in the **final** directory in the **R Programming** directory found on the user **Desktop**.

R Scripts

R Scripts in the **R_Scripts** directory in the **final** directory in the **R Programming** directory found on the user **Desktop**.

I created five objects: Countries, Itemdata, TimeSeries, PlotCountry, and PlotCountryChange. The parameters, details, and return object is listed in each individual R Script.

configuration.R

```
library(dplyr)
library(ggplot2)
library(reshape2)
library(tfplot)

fao_csv <- "../Data/FAO.csv"

source("Countries.R")
source("Itemdata.R")
source("TimeSeries.R")
source("PlotCountry.R")
source("PlotCountryChange.R")
source("Data.R")
```

Data.R

```
load_fao_data <- function()
{
  df <- read.csv(fao_csv, header=T, stringsAsFactors = F)
  return(df)
}

df <- load_fao_data()
```

Countries.R

```
#' Construct of the Countries object
#'
#' @param data.frame eg, df
#'
#' @details A Countries object is a data.frame with
#' columns of Area Abbreviation, Area Code,
#' and Area.
#'
#'Area Abbreviation is the abbreviaton
#' of the Country name, Area Code is the number
#' assigned to a Country, and Area is the
#' name of the country.
#'
#' df is the path file for the data.
#'
#' @return Countries object


Countries<- function(df)
{
  filt_df <- dplyr::select(df, Area.Abbreviation, Area.Code, Area)
  filt_df <- unique(filt_df)
  return(filt_df)
}

get_num.Countries <- function(cy)
{
  num<-length(cy$Area.Code)
  return(num)
}

get_CountryCode.Countries<- function(cy, countryname)
{
  filt_row<-cy[grepl(countryname, cy$Area), ]
  browser()
  x <- filt_row$Area.Code
  return(x)
}
```

Itemdata.R

```
#' Construct of the Itemdata object
#'
```

```
#' @param data.frame eg, df
#'
#' @details A Itemdata object is a data.frame with
#' columns of Items, Item codes,
#' Elements, and Element codes.
#'
#' An Item is the name of the agriculture
#' product produced, Item code is the number
#' assigned to an Item, Element is the type
#' of agriculture product (Food is an
#' agriculture product produced for human
#' consumption and Feed is produced for
#' animal consumption), Element code is
#' the number assigned to an Element.
#'
#' df is the path file for the data.
#'
#' @return Itemdata object

Itemdata<- function(df)
{
  filt_df <- dplyr::filter(df, Area.Code== "41")
  filt_df <- dplyr::select(filt_df, Item, Item.Code, Element, Element.Code)
  return(filt_df)
}

get_Items.Itemdata <- function(id)
{
  return(unique(id$Item))
}

get_num.Itemdata <- function(id)
{
  num<-length(unique(id$Item))
  return(num)
}

get_ItemCode.Itemdata <- function(id, item)
{
  filt_row<-id[grepl(item, id$Item), ]
  x <- filt_row$Item.Code
  return(unique(x))
}

get_ElementCode.Itemdata <- function(id, element)
```

```
{
  filt_row<-id[grepl(element, id$Element), ]
  x <- filt_row$Element.Code
  return(unique(x))
}
```

TimeSeries.R

```
#' Construct of the TimeSeries object
#'
#' @param df data.frame eg, df
#' @param countrycode string eg, "41"
#' @param element string eg, "Food"
#' @param itemcode vector eg, c("2511","2518")
#'
#' @details A TimeSeries object is a time series object
#' with production outputs for each year for every Item from
#' the years 1961 to 2013.
#'
#' @return TimeSeries object

TimeSeries<- function(df, countrycode, element, itemcode)
{
  filt_df <- dplyr::filter(df, Area.Code== countrycode)
  filt_df <- dplyr::filter(filt_df, Element == element)
  filt_df <- dplyr::filter(filt_df, Item.Code %in% itemcode)
  food_matrix <- dplyr::select(filt_df, -Area, -Area.Abbreviation,
                               -Area.Code, -Item.Code, -Element,
                               -Element.Code, -Unit, -latitude, -longitude)
  rev_food_matrix <- setNames(data.frame(t(food_matrix[,-1])), food_matrix[,1])
  #^I did this to retain the Item names for each column
  #after I transposed the data frame.
  tseries <- ts(rev_food_matrix, start = 1961, end=2013, frequency = 1)
  return(tseries)
}
```

PlotCountry.R

```
#' Construct of the PlotCountry object
#'
#' @param df data.frame eg, df
#' @param countrycode string eg, "41"
#' @param element string eg, "Food"
#' @param itemcode vector eg, c("2511","2518")
#'
#' @details A Plot Country object is a melted dataframe
#' that allows for easy plotting using ggplot2
#' with columns Year, variable, and value.
#'
#' Year are the years from 1961 to 2013,
#' variable is the Item name, and value is
#' the recorded production output for a given year.
#'
#' @return PlotCountry object

PlotCountry<- function(df, countrycode, element, itemcode)
{
  filt_df <- dplyr::filter(df, Area.Code== countrycode)
  filt_df <- dplyr::filter(filt_df, Element == element)
  filt_df <- dplyr::filter(filt_df, Item.Code %in% itemcode)

  food_matrix <- dplyr::select(filt_df, -Area, -Area.Abbreviation,
                               -Area.Code, -Item.Code, -Element,
                               -Element.Code, -Unit, -latitude, -longitude)
  rev_food_matrix <- setNames(data.frame(t(food_matrix[,-1])), food_matrix[,1])
  new_df<-data.frame(rev_food_matrix)
  new_df$Year<-(1961:2013)
  #I needed to add a Year column because the melt function
  #melts the data frame so that each row
  #is a unique id-variable combination,
  #I wanted the id to be the Year.
  newnew_df<- new_df %>% select(Year, everything())
  meltdf <- melt(newnew_df,id="Year")
  return(meltdf)
}
```

PlotCountryChange.R

```
#' Construct of the PlotCountryChange object
#'
#' @param ts time series object eg, staples1tsfood
#'
#' @details A PlotCountryChange object is a melted dataframe
#' that allows for easy plotting using ggplot2
#' with columns Year, variable, and value.
#'
#' Year are the years from 1961 to 2013,
#' variable is the Item name, and value is
#' the annual percent change in production.
#'
#' @return PlotCountry object


PlotCountryChange<- function(ts)
{
  pc<- percentChange(ts)
  #I used the function in the tfplot package to calculate
  #percent change for production outputs. The package takes a
  #time series vector or matrix.
  df<-data.frame(pc)
  df$Year<-(1962:2013)
  new_df<- df %>% select(Year, everything())
  meltdf <- melt(new_df,id="Year")

  return(meltdf)
}
```