# This script extracts the ERA5-Land daily mean temperature (deg Kelvin)

# from Google Earth Eng (GEE)

# https://developers.google.com/earth-engine/datasets/catalog/ECMWF\_ERA5\_LAND\_DAILY\_AGGR

# Task : Using sample coordinates of locations (from excel file),

# extract the time series of the required variable for a user

# defined period 01/01/2021 to 31/12/2022.

# If your time series is longer, or you have more locations, GEE can give

# an error related to timeout as the requested data is quite big. The script

# can be adapted to use a for loop to download/extract and combine a longer

# time series of data.

# Required variables for this exercise from ERA5-Land:

# mean 2m temperature [in deg C]

# The corresponding variables on GEE can be found in band section are:

# temperature\_2m [deg K]

# A full list of variables can be found on the dataset page

# https://developers.google.com/earth-engine/datasets/catalog/ECMWF\_ERA5\_LAND\_DAILY\_AGGR

# Last updated 27-Sept-2023 - @Author: Malcolm N. Mistry

# Install and load required packages. For installation of rgee,

# see https://github.com/r-spatial/rgee and run the script\_configure\_rgee.R

# Install required packages before loading

library(sf) # Simple Features for R (Main geo-spatial package) https://r-spatial.github.io/sf/

# library(terra) # Another handy package for raster/vector data, though not used here

library(mapview) # For plotting interactive maps

library(geodata) # To download country shape files with different admin (regions) boundaries

library(readxl) # To read excel sheet

library(tidyr) # Data cleaning, tidying

library(dplyr) # Data cleaning, tidying

library(tictoc) # To time the operation

library(rgee) # For working with GEE API

# Or

# pacman::p\_load(sf, tidyr, tictoc, dplyr,

# mapview, readxl, geodata, rgee)

ee\_check() # check if RGEE works correctly

# Use your gmail account!! Below is my unive.it account which provides access to google drive.

ee\_Initialize("malcolm.mistry@unive.it", drive = TRUE, gcs = FALSE)

# Set path to input directory containing the excel file (locns) and

# output directory where the final RDS files for each variable will be saved.

input\_dir <- '/home/lshmm22/Projects/RGEE/input\_data/'

setwd(input\_dir)

# Select the required ERA5-Land variable (see The image collection info on GEE page)

req\_variable <- "temperature\_2m"

sample\_locns <- read\_excel("R\_users\_group\_seminar\_29Sept2023\_sample\_locns.xlsx",

 sheet = "locns")

head(sample\_locns)

## To use the location coordinates for extracting climate data, we need

## to convert the coordinates to spatial object (here vector)

## Many ways to do it in R. Recommended way is to use package 'sf'.

## Note how we retain the 'location' column as well.

# First, we need to define a Projection/CRS

# Recall that the Earth is NOT FLAT !! Actually, it is not a perfect sphere either!

# proj\_crs <- "+proj=lonlat +datum=WGS84 +no\_defs +ellps=WGS84 +towgs84=0,0,0"

proj\_crs <- 4326

sample\_locns\_sf <- st\_as\_sf(x = sample\_locns,

 coords = c("lon", "lat"),

 crs = proj\_crs)

mapview::mapview(sample\_locns\_sf, legend = FALSE)

## Next step is important!

## Move that geometry from local to earth engine.

ee\_sample\_locns\_sf <- sf\_as\_ee(sample\_locns\_sf)

### Define parameters that will be used for extracting data from GEE

start\_year <- 2021

end\_year <- 2022

startDate <- paste0(as.character(start\_year),"-01-01", sep='') ## format 2018-01-01

# to ensure that last date 31st Dec is included, use last date

# as endDate+1-01-01 (i.e., the 1st of Jan of the following year)

endDate <- paste0(as.character(end\_year+1),"-01-01", sep='')

print(paste('Start date is: ', startDate))

print(paste('End date is: ', endDate))

## Load (Import) ERA5-Land ImageCollection

## Change from image collection to image. You can make use of the

## of the $ syntax instead of the piping operator %>% (both work)

era5\_land\_image\_collection <- ee$ImageCollection("ECMWF/ERA5\_LAND/DAILY\_RAW")$

 filterDate(startDate, endDate)

# OR

# era5\_land\_image\_collection <- ee$ImageCollection("ECMWF/ERA5\_LAND/DAILY\_RAW") %>%

# ee$ImageCollection$filterDate(startDate, endDate)

# Print information about the image collection

# takes a few seconds to print the output

ee\_print(era5\_land\_image\_collection)

era5\_land\_image\_tmean <- era5\_land\_image\_collection$select("temperature\_2m")

# Print information about the image (takes a few seconds to print the output)

ee\_print(era5\_land\_image\_tmean)

# Calculate the nominal scale in meters. This is the native spatial res of

# ERA5-Land and will be used as a parameter later in the below function call

era5\_land\_res <- era5\_land\_image\_tmean$first()$projection()$nominalScale()$getInfo()

# Or easier way, go to https://developers.google.com/earth-engine/datasets/catalog/ECMWF\_ERA5\_LAND\_DAILY\_AGGR#description

# and see under the tab 'Band'

era5\_land\_res <- 11132

## Extract using ee\_extract. Note that the function requires another argument

# 'fun' (see documentation). Here we choose mean, but in principal, the mean

# will do nothing because for each point location the ee\_extract will end up

# extracting only a single grid-cell value over which the location coordinate

# falls. Using the 'era5\_land\_res' as scale argument ensures that the native

# res of era5-land is not changed while extracting the locations' time series

tic()

tmean\_era5\_land\_df <- ee\_extract(x = era5\_land\_image\_tmean,

 y = ee\_sample\_locns\_sf,

 fun = ee$Reducer$mean(),

 scale = era5\_land\_res,

 sf=FALSE)

toc() # Approx 2-3 mins

# Notice that the number of features is 4. Why?

# Because we have 4 locations in our data, so GEE considers these

# 4 coordinates (sf points) as features

class(tmean\_era5\_land\_df)

# Set colnames to correspond to the dates (as X\_yymmdd or as required)

col\_names <- sprintf("X%02s", seq(from=as.Date(paste0(start\_year,"/01/01", sep="")),

 to=as.Date(paste0(end\_year,"/12/31", sep="")),

 by = "day"))

col\_names <- gsub("-", "", col\_names, fixed=TRUE)

col\_names

colnames(tmean\_era5\_land\_df)[2:ncol(tmean\_era5\_land\_df)] <- col\_names

# Now we need to reshape from wide to long format so that the rows in each DF

# are the timeseries (variable value in each day),

# and the columns are the locations.

# This will then help to combine the DFs using rowbind.

tmean\_era5\_land\_df <- tidyr::gather(tmean\_era5\_land\_df,

 date, value, c(2:ncol(tmean\_era5\_land\_df)),

 factor\_key=TRUE)

# Sort by locations and round the mean temperature to two signif digits.

tmean\_era5\_land\_df <- tmean\_era5\_land\_df %>% dplyr::arrange(location)

## The variable is near-surface daily mean Temperature in degrees Kelvin.

## Convert from deg K to deg C. And round it to two significant digits

tmean\_era5\_land\_df$value <- tmean\_era5\_land\_df$value - 273.15

tmean\_era5\_land\_df$value <- round(tmean\_era5\_land\_df$value, digits = 2)

# Or in one line

# tmean\_era5\_land\_df$value <- round(tmean\_era5\_land\_df$value - 273.15,

# digits = 2)

# Rename the third column

names(tmean\_era5\_land\_df)[3] <- c('tmean\_degC')

# Confirm no missing records

apply(is.na(tmean\_era5\_land\_df), 2, which)

# Change date column to yyyy-mm-dd format and class date.

tmean\_era5\_land\_df$date <- sub('.', '', tmean\_era5\_land\_df$date)

# Convert the character string (date column) to date format. The format

# argument is that of the input not the ouput. Since the input column is

# 20210101 format, we use %Y%m%d

tmean\_era5\_land\_df$date <- as.Date(tmean\_era5\_land\_df$date, format = "%Y%m%d")

str(tmean\_era5\_land\_df)

# You can save the the above DF as .RDS, .csv, .xlsx as required.

########### End of script ###########