```
#Summary:
#Producing country-level annual volumetric changes using JPL data
(non-cropped)

library(tidync)
library(data.table)
library(tidyverse)
library(lubridate)
library(zyp)
library(sf)
library(raster)
library(terra)
library(RColorBrewer)
library(rasterVis)
library(xts)

proj_dir = "~/Dropbox/WB/GRACE_Ensemble/"

#Load the GWS data
#Can pick the z-score or cm-equivalent version of GWS

grace_versions = c("GRACE_GWS_2002_2020_wRunoff_BSL2017.csv",
                   "GRACE_GWS_2002_2020_wRunoff_BSL2020_220529.csv",
                   "GRACE_GWS_2002_2020_wRunoff_BSL2012_220508.csv")


grace =
  #fread('Output/z_score/GRACE_GWS_2002_2017_zscore.csv')
  fread(paste0(proj_dir,
               "GRACE_Data/JPL_Mascons/", grace_versions[3]))

#Source: https://doi.org/10.1038/s43016-021-00429-z
crop_area =
  raster("/Users/tejasvi/Dropbox/Mac/Downloads/
Global_cropland_3km_2019.tif")

grace_grid =
  grace %>% dplyr::select(1:3) %>%
  rasterFromXYZ(crs = crs(crop_area))

#Aggregate the cropped area grid to the grace grid using billinear
interpolation
#Convert to pixel data frame so it can be merged with GRACE grid
crop_area_grace =
  crop_area %>%
  resample(grace_grid) %>%
  rasterToPoints() %>%
  as.data.table() %>%
  dplyr::rename(Per_crop_area = Global_cropland_3km_2019)
```

```r
grace_wcrop = merge(grace, crop_area_grace,
                    by.x = c('lon', 'lat'), by.y = c('x','y'), all.x =
T)

#fwrite(grace_wcrop, 'Output/z_score/GRACE_GWS_02_17_zscore_crop.csv')


#Make sure the merging was proper -- regions with expected high
cropped area
# show up (e.g. India)
grace_test =
  grace_wcrop %>% dplyr::select(lon, lat, Per_crop_area) %>%
  rasterFromXYZ(crs = crs(crop_area)) %>%
  plot()

####################
#Use the country level median estimating script

#Load the data --this time it contains %cropped area (z-score version)
# gws_TS =
#   fread('Output/z_score/GRACE_GWS_02_17_zscore_crop.csv')

#cm-equivalent version need to run the code block above
gws_TS = grace_wcrop
######################################################

#Convert to spatial object
gws_TS_spatial =
  st_as_sf(gws_TS, coords = c("lon", "lat"),
           crs = "+proj=longlat +datum=WGS84 +no_defs")

####Load World Regions
wb_regions =
  st_read("WB_Regions/WB_countries_Admin0_10m.shp") %>%
  dplyr::select(WB_NAME, ISO_A2, ISO_A3, ISO_N3, TYPE) %>%
  filter(TYPE != 'Dependency') %>%
  st_make_valid()

wb_regions_ns =
  wb_regions %>% as.data.table() %>% dplyr::select(-geometry) %>%
distinct()

#Merge country data with GRACE
gws_TS_country =
  gws_TS_spatial %>%
  st_make_valid() %>%
  st_join(wb_regions)

crop_flag = T #This flag helps triggering based on cropped arae
cropped.thresh = T #This flag help determine if we should filter
```

```
regions with >20% cropped area or not

if(crop_flag == T & cropped.thresh == T) {
  gws_TS_country =
    gws_TS_country %>%
    dplyr::filter(Per_crop_area>20)
  output.name = 'JPL_country_level_gws_COMB_annual_crop_BSL2017.csv'
} else if(crop_flag == T & cropped.thresh == F){
  gws_TS_country =
    gws_TS_country %>%
    dplyr::filter(Per_crop_area<20)
  output.name =
'JPL_country_level_gws_COMB_annual_non_crop_BSL2017.csv'
}

#Test to see how the remaining GRACE points look
plot(gws_TS_country['Per_crop_area'], cex = 0.25)

#Get list of countries with atleast 36 points
ag.countries =
  gws_TS_country %>%
  as.data.frame() %>%
  group_by(WB_NAME) %>%
  summarise(count = n()) %>%
  dplyr::mutate(area_flag = ifelse(count > 36, T, F))

gws.country.50 =
  gws_TS_country %>%
  merge(ag.countries, by = 'WB_NAME', all.x = T) %>%
  dplyr::select(-Per_crop_area) %>%
  group_by(WB_NAME) %>%
  summarise_if(is.numeric, median, na.rm = TRUE)

gws.country.25 =
  gws_TS_country %>%
  merge(ag.countries, by = 'WB_NAME', all.x = T) %>%
  dplyr::select(-Per_crop_area) %>%
  group_by(WB_NAME) %>%
  summarise_if(is.numeric, function (x){quantile(x,probs = 0.25, na.rm
= TRUE)})

gws.country.75 =
  gws_TS_country %>%
  merge(ag.countries, by = 'WB_NAME', all.x = T) %>%
  dplyr::select(-Per_crop_area) %>%
  group_by(WB_NAME) %>%
  summarise_if(is.numeric, function (x){quantile(x,probs = 0.75, na.rm
= TRUE)})

gws.50.long =
```

```
  gws.country.50 %>%
  gather(yearmon, gws_median, `2002-04`:`2021-01`) %>%
  as.data.table() %>%
  dplyr::select(-cell_id, -geometry, -count) %>%
  drop_na()

gws.25.long =
  gws.country.25 %>%
  gather(yearmon, gws_25, `2002-04`:`2021-01`) %>%
  as.data.table() %>%
  dplyr::select(-cell_id, -geometry, -count) %>% drop_na()

gws.75.long =
  gws.country.75 %>%
  gather(yearmon, gws_75, `2002-04`:`2021-01`) %>%
  as.data.table() %>%
  dplyr::select(-cell_id, -geometry, -count) %>%
  drop_na()

gws.comb =
  merge(gws.50.long, gws.25.long, by = c('WB_NAME','yearmon'), all.x =
T) %>%
  merge(gws.75.long, by = c('WB_NAME','yearmon'), all.x = T) %>%
  merge(wb_regions %>% as.data.frame() %>% dplyr::select(-geometry),
        by = 'WB_NAME', all.x = T) %>%
  dplyr::select(-TYPE) %>%
  merge(ag.countries, by = 'WB_NAME', all.x = T) %>%
  dplyr::mutate(yearmon = as.yearmon(yearmon),
    year = year(yearmon), month = month(yearmon)) %>%
  filter(year<2021)

#Essentially doing the following:
#Obtain a yearly gws value for each country by taking the mean
#For each of the gws columns, obtain the year-by-year difference value
#Using the total cell count, get the estimated volume change

res = 0.5 #resolution in degrees
res_km = res * 111

gws.comb.annual =
  gws.comb %>%
  dplyr::select(WB_NAME, year, gws_median, gws_25, gws_75) %>%
  group_by(WB_NAME, year) %>%
  summarise_if(is.numeric, mean, na.rm = TRUE) %>%
  group_by(WB_NAME) %>%
  mutate(gws_50_diff = gws_median - lag(gws_median),
         gws_25_diff = gws_25 - lag(gws_25),
         gws_75_diff = gws_75 - lag(gws_75)) %>%
  merge(ag.countries, by = 'WB_NAME', all.x = T) %>%
  mutate(gws_50_diff_vol = (gws_50_diff/1e+06) * count * res_km *
```

```
res_km,
         gws_25_diff_vol = (gws_25_diff/1e+06) * count * res_km *
res_km,
         gws_75_diff_vol = (gws_75_diff/1e+06) * count * res_km *
res_km) %>%
  mutate(gws_50_vol = (gws_median/1e+06) * count * res_km * res_km,
         gws_25_vol = (gws_25/1e+06) * count * res_km * res_km,
         gws_75_vol = (gws_75/1e+06) * count * res_km * res_km) %>%
  merge(wb_regions %>% as.data.frame() %>% dplyr::select(-geometry,
-TYPE),
        by = 'WB_NAME', all.x = T)


fwrite(gws.comb.annual,
       paste0(proj_dir,
              "Outputs/JPL_Mascons/",output.name))

######################################################################
########
######################################################################
########
gws.wide.country =
  gws.comb %>%
  dplyr::select(1:3) %>%
  spread(WB_NAME, gws_median)

gws.xts =
  gws.wide.country[, 2:ncol(gws.wide.country)] %>%
  #  dplyr::select(-yearmon) %>%
  xts(as.yearmon(gws.wide.country$yearmon))

gws.smooth =
  rollmean(gws.xts, k = 24) %>%
  as.data.table()


# fwrite(gws.wide.country, 'Country_level_TS/
country_level_gws_MEDIAN.csv')
# fwrite(gws.smooth, 'Country_level_TS/
country_level_gws_MEDIAN_smooth.csv')
# fwrite(gws.comb, 'Output/cm_equivalent/
country_level_gws_centimeter_crop.csv')
fwrite(gws.comb, 'Output/cm_equivalent/country_level_gws_COMB.csv')


#India test

india.st =
  st_read("States/Admin2.shp") %>%
  st_make_valid()
```

```
test =
  gws_TS_country %>%
  filter(WB_NAME=='India') %>%
  st_join(india.st) %>%
  filter(ST_NM=='Punjab')

test.50 =
  test %>%
  summarise_if(is.numeric, median, na.rm = TRUE) %>%
  as.data.frame()

plot(test.50[,3:160])
```