IceSat-2 Mission Orbits

This *first* vignette demonstrates how to download and process *time specific orbits*. We'll download one of the *Reference Ground Track (RGT) cycles* and merge it with other data sources with the purpose to visualize specific areas.

We choose one of the latest which is *"RGT_cycle_14"* (from *December 22, 2021* to *March 23, 2022*) and utilize 8 threads to speed up the pre-processing of the downloaded .kml files (the function takes approximately 15 minutes on my Linux Personal Computer),

```
require(IceSat2R)
require(magrittr)
require(sf)

sf::sf_use_s2(use_s2 = FALSE)                          # disable 's2' in this vignette
mapview::mapviewOptions(leafletHeight = '600px',
                        leafletWidth = '700px')        # applies to all leaflet maps
```

```
avail_cycles = available_RGTs(only_cycle_names = TRUE)
avail_cycles
# [1] "RGT_cycle_1"  "RGT_cycle_2"  "RGT_cycle_3"  "RGT_cycle_4"  "RGT_cycle_5"  "RGT_cycle_6"
#     "RGT_cycle_7"  "RGT_cycle_8"  "RGT_cycle_9"  "RGT_cycle_10" "RGT_cycle_11" "RGT_cycle_12"
#     "RGT_cycle_13" "RGT_cycle_14" "RGT_cycle_15"

choose_cycle = avail_cycles[14]
choose_cycle
# [1] "RGT_cycle_14"


res_rgt_many = time_specific_orbits(RGT_cycle = choose_cycle,
                                    download_method = 'curl',
                                    threads = 8,
                                    verbose = TRUE)
```

We'll then create a data.table based on the *coordinates*, *Date*, *Day of Year*, *Time* and *RGT*,

```
rgt_subs = sf::st_coordinates(res_rgt_many)
colnames(rgt_subs) = c('longitude', 'latitude')
rgt_subs = data.table::data.table(rgt_subs)
rgt_subs$day_of_year = res_rgt_many$day_of_year
rgt_subs$Date = as.Date(res_rgt_many$Date_time)
rgt_subs$hour = lubridate::hour(res_rgt_many$Date_time)
rgt_subs$minute = lubridate::minute(res_rgt_many$Date_time)
rgt_subs$second = lubridate::second(res_rgt_many$Date_time)
rgt_subs$RGT = res_rgt_many$RGT
```

```
res_rgt_many = sf::st_as_sf(x = rgt_subs, coords = c('longitude', 'latitude'), crs = 4326)
res_rgt_many
```

```
## Simple feature collection with 131765 features and 6 fields
## Geometry type: POINT
```

```
## Dimension:     XY
## Bounding box:  xmin: -179.9986 ymin: -87.66742 xmax: 179.9984 ymax: 87.3305
## CRS:           EPSG:4326
## First 10 features:
##    day_of_year       Date hour minute second RGT                      geometry
## 1          356 2021-12-22    7     57     49   1 POINT (-0.1318472 0.02795893)
## 2          356 2021-12-22    7     58     49   1   POINT (-0.5162124 3.868758)
## 3          356 2021-12-22    7     59     49   1    POINT (-0.901809 7.709809)
## 4          356 2021-12-22    8      0     49   1    POINT (-1.289879 11.55065)
## 5          356 2021-12-22    8      1     49   1    POINT (-1.681755 15.39082)
## 6          356 2021-12-22    8      2     49   1     POINT (-2.078916 19.2299)
## 7          356 2021-12-22    8      3     49   1    POINT (-2.483051 23.06748)
## 8          356 2021-12-22    8      4     49   1    POINT (-2.896146 26.90316)
## 9          356 2021-12-22    8      5     49   1      POINT (-3.3206 30.73662)
## 10         356 2021-12-22    8      6     49   1    POINT (-3.759374 34.56754)
```

Icesat-2 and Countries intersection

We'll proceed to merge the orbit geometry points with the countries data of the *rnaturalearth* R package (1:110 million scales) and for this purpose, we keep only the *"sovereignt"* and *"sov_a3"* columns,

```
cntr = rnaturalearth::ne_countries(scale = 110, type = 'countries', returnclass = 'sf')
cntr = cntr[, c('sovereignt', 'sov_a3')]
cntr
```

```
## Simple feature collection with 177 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -90 xmax: 180 ymax: 83.64513
## CRS:           +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## First 10 features:
##             sovereignt sov_a3                       geometry
## 0          Afghanistan    AFG MULTIPOLYGON (((61.21082 35...
## 1               Angola    AGO MULTIPOLYGON (((16.32653 -5...
## 2              Albania    ALB MULTIPOLYGON (((20.59025 41...
## 3 United Arab Emirates    ARE MULTIPOLYGON (((51.57952 24...
## 4            Argentina    ARG MULTIPOLYGON (((-65.5 -55.2...
## 5              Armenia    ARM MULTIPOLYGON (((43.58275 41...
## 6           Antarctica    ATA MULTIPOLYGON (((-59.57209 -...
## 7               France    FR1 MULTIPOLYGON (((68.935 -48....
## 8            Australia    AU1 MULTIPOLYGON (((145.398 -40...
## 9              Austria    AUT MULTIPOLYGON (((16.97967 48...
```

We then merge the orbit points with the country geometries and specify also *"left = TRUE"* to keep also observations that do not intersect with the *rnaturalearth* countries data,

```
dat_both = suppressMessages(sf::st_join(x = res_rgt_many,
                                        y = cntr,
                                        join = sf::st_intersects,
                                        left = TRUE))
dat_both
```

```
## Simple feature collection with 131765 features and 8 fields
## Geometry type: POINT
## Dimension:     XY
```

```
## Bounding box:  xmin: -179.9986 ymin: -87.66742 xmax: 179.9984 ymax: 87.3305
## CRS:           EPSG:4326
## First 10 features:
##    day_of_year       Date hour minute second RGT   sovereignt sov_a3
## 1          356 2021-12-22    7     57     49   1         <NA>   <NA>
## 2          356 2021-12-22    7     58     49   1         <NA>   <NA>
## 3          356 2021-12-22    7     59     49   1        Ghana    GHA
## 4          356 2021-12-22    8      0     49   1 Burkina Faso    BFA
## 5          356 2021-12-22    8      1     49   1         Mali    MLI
## 6          356 2021-12-22    8      2     49   1         Mali    MLI
## 7          356 2021-12-22    8      3     49   1         Mali    MLI
## 8          356 2021-12-22    8      4     49   1      Algeria    DZA
## 9          356 2021-12-22    8      5     49   1      Algeria    DZA
## 10         356 2021-12-22    8      6     49   1      Morocco    MAR
##                          geometry
## 1  POINT (-0.1318472 0.02795893)
## 2    POINT (-0.5162124 3.868758)
## 3     POINT (-0.901809 7.709809)
## 4     POINT (-1.289879 11.55065)
## 5     POINT (-1.681755 15.39082)
## 6      POINT (-2.078916 19.2299)
## 7     POINT (-2.483051 23.06748)
## 8     POINT (-2.896146 26.90316)
## 9       POINT (-3.3206 30.73662)
## 10    POINT (-3.759374 34.56754)
```

The unique number of RGT's for *"RGT_cycle_14"* are

```
length(unique(dat_both$RGT))
```

```
## [1] 1387
```

We observe that from *December 22, 2021* to *March 23, 2022*,

```
df_tbl = data.frame(table(dat_both$sovereignt), stringsAsFactors = F)
colnames(df_tbl) = c('country', 'Num_IceSat2_points')

df_subs = dat_both[, c('RGT', 'sovereignt')]
df_subs$geometry = NULL
df_subs = data.table::data.table(df_subs, stringsAsFactors = F)
colnames(df_subs) = c('RGT', 'country')
df_subs = split(df_subs, by = 'country')
df_subs = lapply(df_subs, function(x) {
  unq_rgt = sort(unique(x$RGT))
  items = ifelse(length(unq_rgt) < 5, length(unq_rgt), 5)
  concat = paste(unq_rgt[1:items], collapse = '-')
  iter_dat = data.table::setDT(list(country = unique(x$country),
                                    Num_RGTs = length(unq_rgt),
                                    first_5_RGTs = concat))
  iter_dat
})

df_subs = data.table::rbindlist(df_subs)

df_tbl = merge(df_tbl, df_subs, by = 'country')
df_tbl = df_tbl[order(df_tbl$Num_IceSat2_points, decreasing = T), ]
```

```
DT_dtbl = DT::datatable(df_tbl, rownames = FALSE)
```

Show 10 entries Search:

country	Num_IceSat2_points	Num_RGTs	first_5_RGTs
Antarctica	12007	1387	1-2-3-4-5
Russia	5891	1215	1-2-3-4-5
Canada	3546	665	4-5-6-11-12
United States of America	2265	647	4-5-6-7-12
China	1892	457	3-4-5-11-12
Brazil	1496	374	2-3-10-11-17
Australia	1412	307	2-3-9-10-17
Denmark	1349	407	2-3-10-11-17
Kazakhstan	693	292	5-6-13-14-20
Argentina	562	136	3-18-26-33-41

Showing 1 to 10 of 152 entries Previous 1 2 3 4 5 ... 16 Next

all RGT's (1387 in number) intersect with *"Antarctica"* and almost all with *"Russia"*.

'Onshore' and Offshore Points IceSat-2 coverage

The **onshore** and **offshore** number of IceSat-2 points and percentages for the *"RGT_cycle_14"* equal to

```
num_sea = sum(is.na(dat_both$sovereignt))
num_land = sum(!is.na(dat_both$sovereignt))

perc_sea = round(num_sea / nrow(dat_both), digits = 4) * 100.0
perc_land = round(num_land / nrow(dat_both), digits = 4) * 100.0

dtbl_land_sea = data.frame(list(percentage = c(perc_sea, perc_land),
                                Num_Icesat2_points = c(num_sea, num_land)))

row.names(dtbl_land_sea) = c('sea', 'land')
```

```
stargazer::stargazer(dtbl_land_sea,
                     summary = FALSE,
                     rownames = FALSE,
                     header = FALSE,
                     table.placement = 'h',
                     title = 'Land and Sea Proportions')
```

Table 1: Land and Sea Proportions

percentage	Num_Icesat2_points
67.070	88,369
32.930	43,396

Global glaciated areas and IceSat-2 coverage

We can also observe the IceSat-2 *"RGT_cycle_14"* coverage based on the 1 to 10 million large scale Natural Earth Glaciated Areas data,

```
ne_glaciers = system.file('data_files', 'ne_10m_glaciated_areas.RDS', package = "IceSat2R")
ne_obj = readRDS(file = ne_glaciers)
```

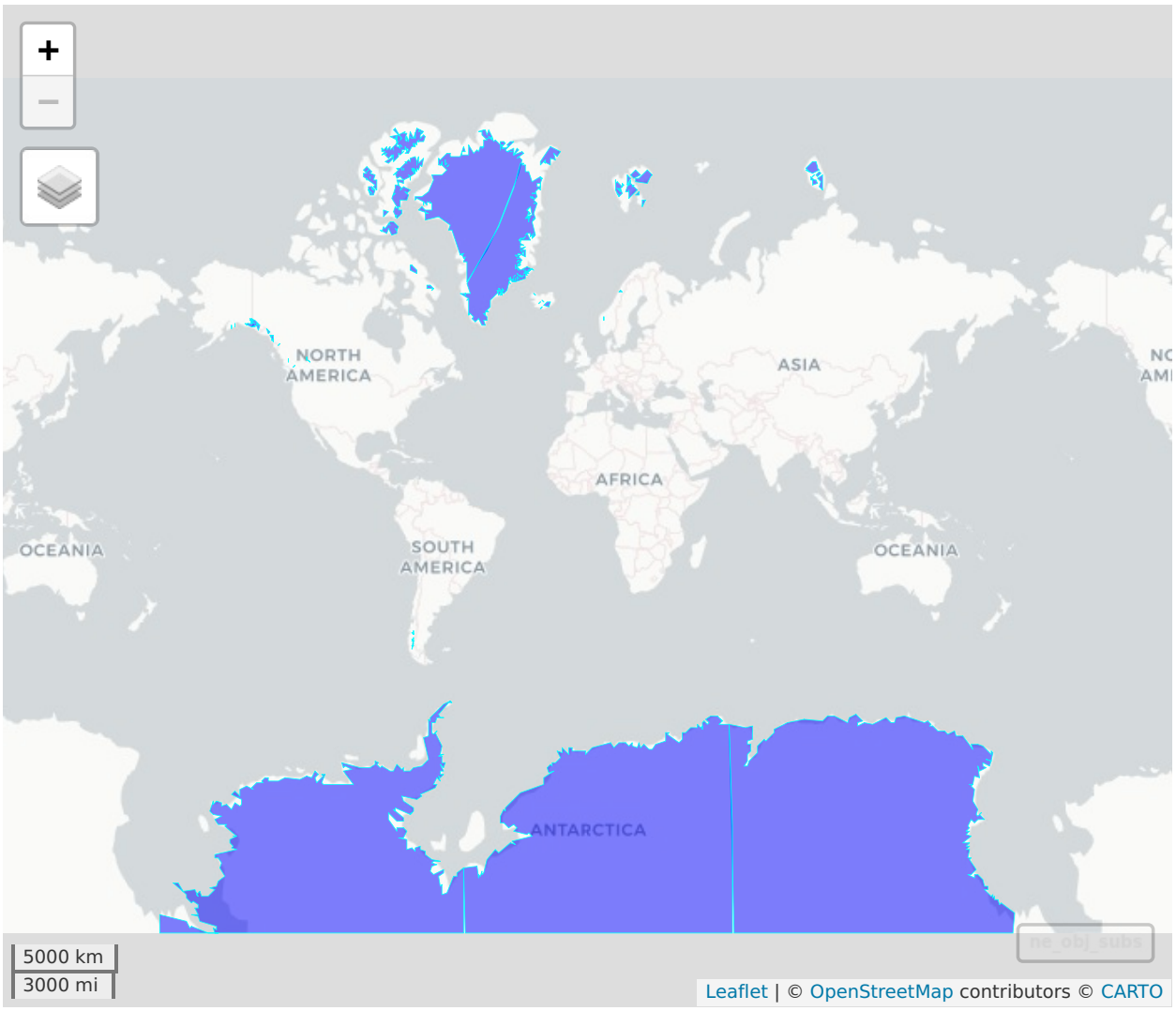
We'll restrict the processing to the major polar glaciers (that have a name included),

```
ne_obj_subs = subset(ne_obj, !is.na(name))
ne_obj_subs = sf::st_make_valid(x = ne_obj_subs)      # check validity of geometries
ne_obj_subs
```

```
## Simple feature collection with 68 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -89.99993 xmax: 180 ymax: 82.96573
## CRS:           4326
## First 10 features:
##     recnum scalerank      featurecla                    name min_zoom
## 143    143         3 Glaciated areas    Mount Brown Icefield      2.1
## 148    148         5 Glaciated areas    Braithwaite Icefield      5.0
## 152    152         3 Glaciated areas         Hooker Icefield      2.1
## 206    206         5 Glaciated areas       Homathko Icefield      5.0
## 214    214         6 Glaciated areas Clachnacudainn Icefield      5.7
## 215    215         6 Glaciated areas         Albert Icefield      5.7
## 228    228         3 Glaciated areas        Plateau Icefield      2.1
## 230    230         5 Glaciated areas      Pemberton Icefield      5.0
## 256    256         3 Glaciated areas        Cambria Icefiled      2.1
## 273      0         3 Glaciated areas          Lyell Icefield      2.1
##                           geometry
## 143 MULTIPOLYGON (((-118.4066 5...
## 148 MULTIPOLYGON (((-119.9303 5...
## 152 MULTIPOLYGON (((-117.8572 5...
## 206 MULTIPOLYGON (((-124.6489 5...
## 214 MULTIPOLYGON (((-118.0284 5...
## 215 MULTIPOLYGON (((-117.6752 5...
## 228 MULTIPOLYGON (((-123.8453 5...
## 230 MULTIPOLYGON (((-123.3869 5...
## 256 MULTIPOLYGON (((-129.661 56...
## 273 MULTIPOLYGON (((-117.2649 5...
```

and we'll visualize the subset using the *mapview* package,

```
mpv = mapview::mapview(ne_obj_subs,
                       color = 'cyan',
                       col.regions = 'blue',
                       alpha.regions = 0.5,
                       legend = FALSE)
mpv
```



We will see which orbits of the IceSat-2 *"RGT_cycle_14"* intersect with these major polar glaciers,

```
res_rgt_many$id_rgt = 1:nrow(res_rgt_many)       # include 'id' for fast subsetting

dat_glac_sf = suppressMessages(sf::st_join(x = ne_obj_subs,
                                           y = res_rgt_many,
                                           join = sf::st_intersects))

dat_glac = data.table::data.table(sf::st_drop_geometry(dat_glac_sf), stringsAsFactors = F)
dat_glac = dat_glac[complete.cases(dat_glac), ]              # keep non-NA observations
dat_glac
```

```
##         recnum scalerank      featurecla            name min_zoom day_of_year
##      1:    952         4 Glaciated areas  Jostedalsbreen      3.0          40
##      2:   1696         3 Glaciated areas Agassiz Ice Cap      2.1         357
##      3:   1696         3 Glaciated areas Agassiz Ice Cap      2.1         358
##      4:   1696         3 Glaciated areas Agassiz Ice Cap      2.1         361
##      5:   1696         3 Glaciated areas Agassiz Ice Cap      2.1         362
##     ---
## 13245:      0         3 Glaciated areas  Kluane Ice Cap      2.1          42
```

```
## 13246:      0         3 Glaciated areas  Kluane Ice Cap      2.1          44
## 13247:      0         3 Glaciated areas  Kluane Ice Cap      2.1          48
## 13248:      0         3 Glaciated areas  Kluane Ice Cap      2.1          71
## 13249:      0         3 Glaciated areas  Kluane Ice Cap      2.1          73
##              Date hour minute second  RGT id_rgt
##     1: 2022-02-09   17     23     15  755  71662
##     2: 2021-12-23    1     41      0   12   1072
##     3: 2021-12-24   12     10     22   34   3157
##     4: 2021-12-27    1     32     40   73   6867
##     5: 2021-12-28   12      2      3   95   8952
##    ---
## 13245: 2022-02-11   14     42     39  784  74402
## 13246: 2022-02-13    3      6     19  807  76602
## 13247: 2022-02-17    2     57     59  868  82397
## 13248: 2022-03-12   13     18     42 1226 116392
## 13249: 2022-03-14    1     42     22 1249 118592
```

We'll split the merged data by the *'name'* of the glacier,

```
dat_glac_name = split(x = dat_glac, by = 'name')

sum_stats_glac = lapply(dat_glac_name, function(x) {

  dtbl_glac = x[, .(name_glacier = unique(name),
                    Num_unique_Dates = length(unique(Date)),
                    Num_unique_RGTs = length(unique(RGT)))]
  dtbl_glac
})

sum_stats_glac = data.table::rbindlist(sum_stats_glac)
sum_stats_glac = sum_stats_glac[order(sum_stats_glac$Num_unique_RGTs, decreasing = T), ]
```

The next table shows the total number of days and RGTs for each one of the major polar glaciers,

```
stargazer::stargazer(sum_stats_glac,
                     summary = FALSE,
                     rownames = FALSE,
                     header = FALSE,
                     table.placement = 'h',
                     title = 'Days and RGTs')
```

We can restrict to one of the glaciers to visualize the IceSat-2 *"RGT_cycle_14"* coverage over this specific area (*'Southern Patagonian Ice Field'*),

```
sample_glacier = 'Southern Patagonian Ice Field'
dat_glac_smpl = dat_glac_name[[sample_glacier]]
```

```
cols_display = c('name', 'day_of_year', 'Date', 'hour', 'minute', 'second', 'RGT')

stargazer::stargazer(dat_glac_smpl[, ..cols_display],
                     summary = FALSE,
                     rownames = FALSE,
                     header = FALSE,
                     table.placement = 'h',
                     title = 'Southern Patagonian Ice Field')
```

and we gather the intersected RGT coordinates points with the selected glacier,

Table 2: Days and RGTs

name_glacier	Num_unique_Dates	Num_unique_RGTs
Antarctic Ice Sheet	92	1,387
Greenland Ice Sheet	91	352
Agassiz Ice Cap	56	58
Academy of Sciences Ice Cap	34	34
Manson Icefield	14	19
Müller Ice Cap	16	16
Kluane Ice Cap	12	12
Sydkap Ice Cap	6	7
Southern Patagonian Ice Field	5	5
Stikine Icecap	4	4
Vestfonna	3	3
Brasvellbreen	3	3
Northern Patagonian Ice Field	2	2
Jostedalsbreen	1	1

Table 3: Southern Patagonian Ice Field

name	day_of_year	Date	hour	minute	second	RGT
Southern Patagonian Ice Field	357	2021-12-23	0	40	43	11
Southern Patagonian Ice Field	2	2022-01-02	12	28	4	171
Southern Patagonian Ice Field	20	2022-01-20	23	16	46	453
Southern Patagonian Ice Field	49	2022-02-18	21	52	48	895
Southern Patagonian Ice Field	64	2022-03-05	9	31	50	1,116

```
subs_rgts = subset(res_rgt_many, id_rgt %in% dat_glac_smpl$id_rgt)

set.seed(1)
samp_colrs = sample(x = grDevices::colors(distinct = TRUE),
                    size = nrow(subs_rgts))
subs_rgts$color = samp_colrs
```

```
ne_obj_subs_smpl = subset(ne_obj_subs, name == sample_glacier)

mpv_glacier = mapview::mapview(ne_obj_subs_smpl,
                               color = 'cyan',
                               col.regions = 'blue',
                               alpha.regions = 0.5,
                               legend = FALSE)

mpv_RGTs = mapview::mapview(subs_rgts,
                            color = subs_rgts$color,
                            alpha.regions = 0.0,
                            lwd = 6,
                            legend = FALSE)
```

and visualize both the glacier and the subset of the intersected RGT coordinate points (of the different Days) in the same map. The clickable map and point popups include more information,

```
lft = mpv_glacier + mpv_RGTs
lft
```