The Journal of Open Source Software

ICESat-2 Altimeter Data using R

Lampros Sp. Mouselimis[1]

1 Monopteryx, Rahouli Paramythias, Thesprotia, Greece

DOI:

Software

- Review
- Repository
- Archive

Submitted:
Published:

Summary

Lidar technology (light detection and ranging) is integrated into autonomous cars, unmanned aerial vehicles (UAV), airplanes, and satellites and it represents a growing market. Many satellite missions utilize lidar to observe the surface of the earth and one of these is ICESat-2 (Markus et al., 2017), which was launched on 15th September 2018 with the primary goal to measure changes in glaciers and ice sheets (Smith et al., 2019). ICESat-2 (the successor of ICESat) consists of 6 beams (3 pairs), where each pair is separated by 3 kilometers and each beam in the pair is also separated by 90 meters as the following image shows (Smith et al., 2019),

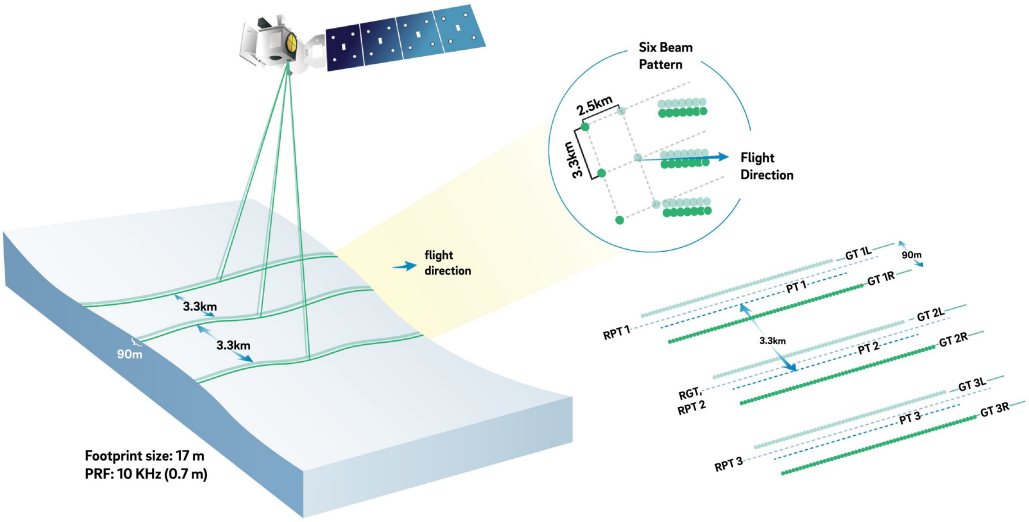

Figure 1: the ICESat-2 ATLAS six-beam pattern

Besides the Distributed Active Archive Center (DAAC) at NSIDC another source to retrieve ICESat-2 data is the OpenAltimetry platform, which allows users to discover, access, and visualize data from NASA's ICESat and ICESat-2 missions (Khalsa et al., 2020).

The IceSat2R package (Mouselimis, 2022) includes functionality that

- creates a connection to the OpenAltimetry Web API
- allows users to interactively create a global degree-grid, which is required for the OpenAltimetry queries
- makes feasible the download of the required Reference Ground Tracks (RGTs) from the National Snow & Ice Data Center (NSIDC)
- includes three vignettes that explain in detail how users can analyze ICESat-2 data and combine it with other satellite and in-situ sources.

Statement of need

Online access to data and analysis tools via easy-to-use interfaces can significantly increase data usage across a wide range of users (Khalsa et al., 2020). In the same way, authors of programming packages should incorporate the required functionality so that the biggest possible number of users can access and take advantage of the codebase. An important aspect of the IceSat2R package is that it includes the code, documentation, and examples so that users can retrieve, process, and analyze data based on specific workflows. For instance,

- A user can select an *area of interest* (AOI) either programmatically or interactively
- If the *Reference Ground Track* (RGT) is not known, the user has the option to utilize either
  - one of the **'overall_mission_orbits()'** or **'time_specific_orbits()'** to compute the RGT(s) for a pre-specified global area or for a time period, or
  - one of the **'vsi_nominal_orbits_wkt()'** or **'vsi_time_specific_orbits_wkt()'** to compute the RGT(s) for a specific AOI
- Once the RGT is computed it can be verified with the **'getTracks()'** function of the OpenAltimetry Web API
- Finally the user can utilize one of the **'get_atlas_data()'** or **'get_level3a_data()'** functions to retrieve the data for specific product(s), Date(s) and Beam(s)

This work-flow is illustrated also in the next diagram,

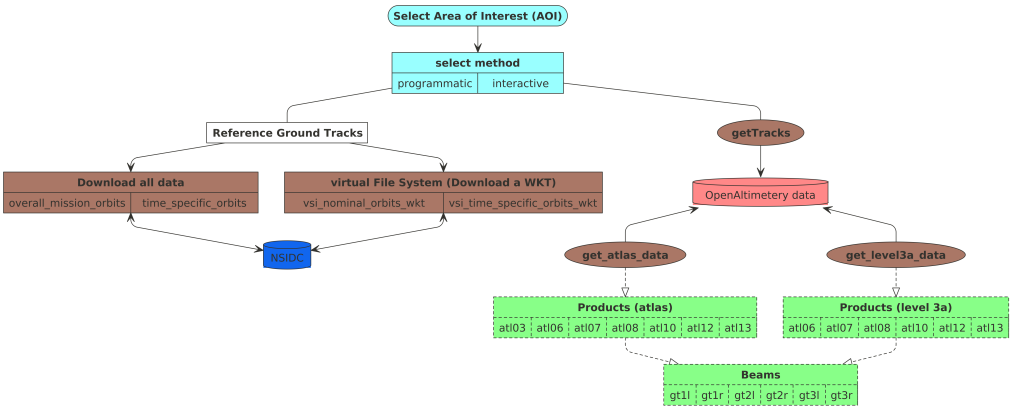

Figure 2: IceSat2R Workflow Diagram

State of the field

In R programming there isn't currently a CRAN package that allows a user to download ICESat-2 data for the majority of the data products.

- The icesat2R R package (Queinnec, 2022) appears only on Github and based on the DESCRIPTION file it includes code that allows the "Downloading and processing of ICESat-2 data for forest analysis" (Atlas-03 and Atlas-08 Products).
- In Python programming the icepyx (Scheick & others, 2019--) is both a software library and a community composed of ICESat-2 data users, developers, and the scientific community that work together to develop a shared library of resources that

simplify the process of querying, obtaining, analyzing, and manipulating ICESat-2 datasets to enable scientific discovery.
- In Julia programming the SpaceLiDAR.jl (Pronk, 2021) is a research package that currently supports the "GLAH14 v34," "ATL03 v4," "ATL08 v4" and "ATL12 v4" ICESat-2 data products.

The main difference between "IceSat2R" and the other programming packages ("icesat2R," "icepyx," "SpaceLiDAR.jl") is that the "IceSat2R" package allows the users to download ICESat-2 data using the OpenAltimetry API whereas users of the remaining software download ICESat-2 data directly from NSIDC.

Example use case

The example code of this section explains how an R user can utilize the IceSat2R package to,

- extract the *Reference Ground Tracks (RGT)* of an AOI (*Himalayas mountain range*) using the *GDAL Virtual File System*
- retrieve data from the *OpenAltimetry API* for a pre-specified bounding box (as described in the previous work-flow)
- report the number of downloaded and available observations for the specified ICESat-2 product

The following map shows the bounding box areas (small on the left and big on the right) that will be used in the code.

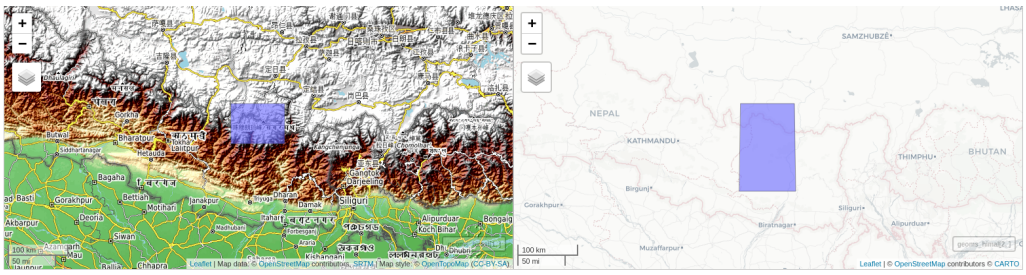

Figure 3: Himalayas AOI

```
# load the required R packages and the AOI

pkgs = c('IceSat2R', 'magrittr', 'sf', 'mapview', 'leaflet',
         'glue', 'data.table', 'dplyr')

load_pkgs = lapply(pkgs, require, character.only = TRUE)

geoms_himal = system.file('data_files', 'vignette_data', 'himalayas.RDS',
                          package = "IceSat2R") %>% readRDS()

sf_wkt = sf::st_geometry(subset(geoms_himal, area_size == 'big'))
centr_wkt = sf::st_coordinates(sf::st_centroid(sf_wkt))
dat_wkt = sf::st_as_text(sf_wkt)

# set the "rgt_repeat" to 1 (there is also the option to iterate over the
# 8 available repeats for the Eastern Hemisphere and extract the unique RGTs

unq_rgts = IceSat2R::vsi_nominal_orbits_wkt(orbit_area = 'eastern_hemisphere',
```

```
                                                track = 'GT7',
                                                rgt_repeat = 1,
                                                wkt_filter = dat_wkt,
                                                download_method = 'curl',
                                                download_zip = FALSE,
                                                verbose = TRUE)[[1]]$RGT
unq_rgts
# [1] "96"   "157"  "363"  "538"  "599"  "805"  "866"  "1041" "1308"
```

In this use case, we are interested in ICESat-2 data for a specific time period (from *'2020-01-01'* to *'2021-01-01'* - 1-year's data). We'll make use of the *IceSat2R::vsi_time_specific_orbits_wkt()* function which queries all *15 ICESat-2 RGTs cycles* (as of March 2022) to come to the RGTs intersection for the specified 1-year time interval,

```
date_start = '2020-01-01'
date_end = '2021-01-01'

orb_cyc_multi = IceSat2R::vsi_time_specific_orbits_wkt(date_from = date_start,
                                                       date_to = date_end,
                                                       RGTs = unq_rgts,
                                                       wkt_filter = dat_wkt,
                                                       verbose = TRUE)
```

The next map shows the 18 different Date-Time matches for our defined 1-year time period based on the output of the *'IceSat2R::vsi_time_specific_orbits_wkt()'* function,

```
orbit_cy = mapview::mapview(orb_cyc_multi, legend = FALSE)
AOI_wkt = mapview::mapview(sf_wkt, legend = FALSE)
lft = orbit_cy + AOI_wkt

lft@map %>% leaflet::setView(lng = centr_wkt[, 'X'],
                             lat = centr_wkt[, 'Y'],
                             zoom = 7)
```

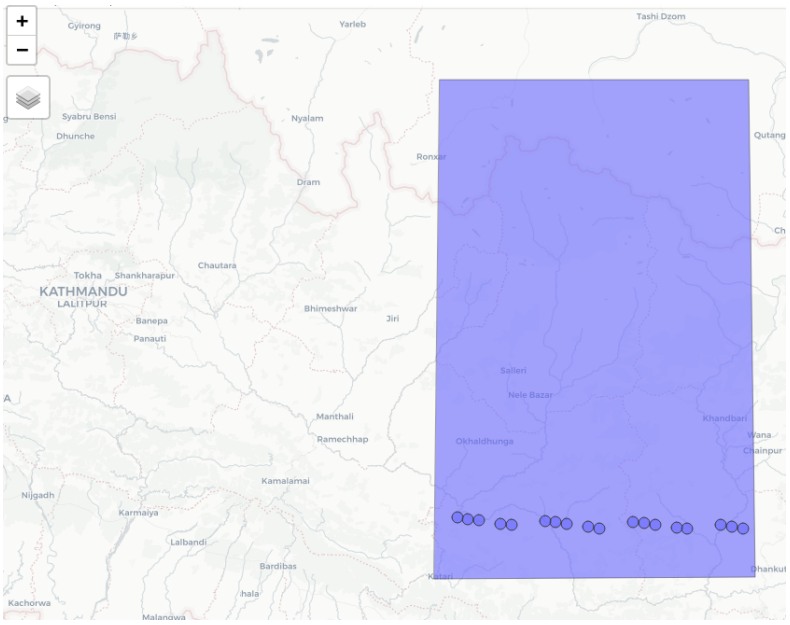

Figure 4: Intersection of the AOI and the ICESat-2 Orbits

The output of *'vsi_time_specific_orbits_wkt()'* can be verified with the *OpenAltimetry's 'getTracks()'* function, so that we can keep the unique RGTs,

```
dtbl_rgts = sf::st_bbox(obj = sf_wkt) %>%
  verify_RGTs(nsidc_rgts = orb_cyc_multi, verbose = TRUE)

RGTs = orb_cyc_multi[dtbl_rgts$RGT_OpenAlt == dtbl_rgts$RGT_NSIDC, ][['RGT']] %>%
  unique()
```

We also restrict our initial AOI to a smaller area in the *Himalayas mountain range*,

```
bbx_aoi_init = sf::st_geometry(subset(geoms_himal, area_size == 'small')) %>%
  sf::st_bbox()
```

A potential use case would be to visualize the *Ice*, *Land*, and *Canopy* height differences, and the right function for this purpose would be the *IceSat2R::get_level3a_data()* function that takes a *time interval* as input. The corresponding ICESat-2 and OpenAltimetry API products are the **'atl06'** and **'atl08'**. In this example use case we'll restrict to the *'atl06'* product (iterating over multiple Products is also feasible), We'll iterate over the available tracks to retrieve the altimeter data,

```
prod = 'atl06'
dat_out = logs_out = list()

for (track_i in RGTs) {

  cat(paste0(track_i, '.'))
  iter_dat = IceSat2R::get_level3a_data(minx = bbx_aoi_init['xmin'],
                                        miny = bbx_aoi_init['ymin'],
                                        maxx = bbx_aoi_init['xmax'],
                                        maxy = bbx_aoi_init['ymax'],
                                        startDate = date_start,
                                        endDate = date_end,
                                        trackId = track_i,
                                        product = prod)
  iter_logs = list(RGT = track_i,
                   Product = prod,
                   N_rows = nrow(iter_dat))

  logs_out[[as.character(track_i)]] = data.table::setDT(iter_logs)
  dat_out[[as.character(track_i)]] = iter_dat
}
```

The following table shows the RGTs and the retrieved number of rows for the specified ICESat-2 products.

```
dtbl_logs = data.table::rbindlist(logs_out) %>%
  subset(N_rows > 0) %>%
  dplyr::arrange(dplyr::desc(N_rows))

#     RGT Product N_rows
# 1:  599   atl06  56531
# 2:   96   atl06  51874
# 3: 1041   atl06  49208
# 4:  538   atl06   1535
```

Acknowledgements

The development of the IceSat2R package was supported by the R Consortium (grant code 21-ISC-2-2).

References

Khalsa, S. J. S., Borsa, A., Nandigam, V., Phan, M., Lin, K., Crosby, C., Fricker, H., et al. (2020). OpenAltimetry - rapid analysis and visualization of Spaceborne altimeter data. *Earth Science Informatics*. doi:10.1007/s12145-020-00520-2

Markus, T., Neumann, T., Martino, A., Abdalati, W., Brunt, K., Csatho, B., Farrell, S., et al. (2017). The ice, cloud, and land elevation satellite-2 (ICESat-2): Science requirements, concept, and implementation. *Remote Sensing of Environment*, *190*, 260–273. doi:https://doi.org/10.1016/j.rse.2016.12.029

Mouselimis, L. (2022). *IceSat2R: ICESat-2 altimeter data using r*. Retrieved from https://CRAN.R-project.org/package=IceSat2R

Pronk, M. (2021). *SpaceLiDAR.jl*. Retrieved from https://github.com/evetion/SpaceLiDAR.jl

Queinnec, M. (2022). Downloading and processing ICESat-2 data for forest analysis. *GitHub repository*. https://github.com/mqueinnec/icesat2R; GitHub.

Scheick, J., & others. (2019--2019--). icepyx: Python tools for obtaining and working with ICESat-2 data. icesat2py. Retrieved from https://github.com/icesat2py/icepyx

Smith, B., Fricker, H. A., Holschuh, N., Gardner, A. S., Adusumilli, S., Brunt, K. M., Csatho, B., et al. (2019). Land ice height-retrieval algorithm for NASA's ICESat-2 photon-counting laser altimeter. *Remote Sensing of Environment*, *233*, 111352. doi:10.1016/j.rse.2019.111352