Working with Sentinel-3 altimetry data via R

Karina Nielsen

2024-01-22

Contents

Preface **2**

WorldWater project background **2**

Installation **2**
Dependencies . . . 2

Getting started with R **2**

Function collection overview **3**

Data available for testing **4**

Part 1 read Sentinel-3 data **4**
Exercise: Reading data . . . 4

Part 2 Extracting data using a shapefile **4**
Exercise: Extracting data via a shapefile . . . 5

Part 3 Generating lake level time series **5**
Exercise: plotting data . . . 5
Exercise: Time series reconstruction . . . 5

Summary of scripts **6**
Putting it all together read and extract data . . . 6
Putting it all together reconstruct time series . . . 6

Preface

This document describes how to obtain water surface elevations from Sentinel-3A/3B satellite altimetry data and how to reconstruct the water level time series. The needed code and a test data set to illustrate the functionality is available from https://github.com/cavios/WorldWater.

WorldWater project background

WorldWater develops cost-effective methods for large-area and high-temporal monitoring of all water bodies (i.e. inland and coastal, lakes/reservoirs and rivers) in both their extent and volume. Surface Water Extent (SWE) is derived from the synergistic usage of optical and SAR imagery and integrating all observations from free and open satellite missions, i.e. Sentinel 1, Sentinel 2 and Landsat. The surface volume is inferred using Water Surface Elevation (WSE) estimates from radar altimetry observations based on Sentinel-3A and 3B. The key points of the project are:

- Development and provision of innovative, yet robust and operational Earth Observation based methods and tools for monitoring surface water dynamics.
- Demonstration of the suitability and scalability of the surface water dynamics products and indicators over selected national and regional demonstration sites.
- Integration of surface water dynamics products and tools in a data analytics platform to foster the wider usage of Earth Observation data for country and basin scale water resource monitoring and reporting.
- Supporting users and improving national capacities to better monitor and report on water resources using Earth Observation through user workshops, training courses and awareness events.

Installation

The provided examples are implemented in the open source software language “R” which can be found at https://www.r-project.org/. To have a graphical user interfase “Rstudio” https://posit.co/download/rstudio-desktop/ can be recommended.
To generate water level time series yo can use the "tsHydro" package which is available via GitHub or via r-universe.

```
install.packages('tsHydro',
                 repos=c(CRAN="https://cloud.r-project.org/",
                 tshydro='https://cavios.r-universe.dev'))
```

Dependencies

In addition to the `tsHydro` R-package some R-packages are needed to rum the code examples which can be installed from R with the function `install.packages`. Hence, to install the package “hdf5r”use the following command for the R terminal:

```
install.packages("hdf5r")
```

- date
- lubridate
- hdf5r
- terra

Getting started with R

This section gives a short introduction to R, which is useful to new R users.

R tutorials can be found at the r-project web side https://cran.r-project.org/manuals.html

Help pages can be accessed by typing "?" in front of a given function. If we want to access the help for the function `plot` we write

```
?plot
```

To start the web based help interface

```
help.start()
```

To exit R write

```
q()
```

Where am I? To get the path of your working directory

```
getwd()
```

To change working directory

```
setwd('path to the new wd')
```

lists the files and directories in the current directory

```
dir()
```

lists the available R objects

```
ls()
```

Function collection overview

The needed code to get started working with Sentinel-3 is available via GitHub https://github.com/cavios/WorldWater/tree/main/Rcode. To assist you in working with Sentinel-3 altimetry data a small set of functions are available in the file `S3Function.R`

With the function `readS3` you can read a Sentinel-3 'standard_measurement.nc' file and generate surface elevations with respect to the geoid model EGM2008. The function has the following arguments:

```
readS3(nc,tofile=FALSE, discardNA=TRUE)
```

- `nc` the Sentinel-3 input file
- `tofile` if TRUE the output is saved to a file (default is FALSE)
- `discardNA` as default NA values are discarded

The function will provide a data set with the following columns

1. `timesec` time in secounds since 01-01-2000
2. `time` time in decimal years (daily resolution)
3. `cycle` cycle number there is one cycle pr 27 days
4. `sattrack` the number of the pass
5. `lat` Latitude in decimal degrees
6. `lon` Longitude in decimal degrees
7. `height` elevation [m] w.r.t. geoid EMG2008
8. `geoid` height [m] of EGM2008

With the function `getInShape` you can extract data within a shapefile e.g. a lake polygon mask. The function has the following arguments:

```
getInShape(dat,myshape,colid=1,doplot=FALSE)
```

- `dat` output from `readS3`
- `myshape` a shapefile to extract data e.g. a lake polygon or a river boundary

- `colid` the column in the shapefile containing information about lake polygon id
- `doplot` if TRUE a plot of the extracted data is generated

Data available for testing

To illustrate the functionality a test data set is made available via GitHub https://github.com/cavios/WorldWater/tree/main/testdata. Here you will find a Sentinel-3 data file and a already extracted data for one lake in the file `lakedata_4610001882.csv`. In addition you can also find a lake polygon shapefile in the folder "mask" https://github.com/cavios/WorldWater/tree/main/mask

Part 1 read Sentinel-3 data

In the following example it is described how to read a Sentinel-3 file.

```
source('S3Functions.R') # read in functions
datadir<-'testdata'     # define data folder in our case it is 'testdata'
myfiles<-dir(datadir,full.names=TRUE,recursive=TRUE,
          pattern='standard_measurement') # create a list with Sentinel-3 files in the folder "testdata

dat<-readS3(myfiles[1],tofile=FALSE, discardNA=TRUE)

head(dat)
```

```
##      timesec     time cycle sattrack      lat      lon     height    geoid
## 11 745046730 2023.609   102      105 81.42873 158.2904 -0.6545536 1.801926
## 12 745046730 2023.609   102      105 81.42871 158.2704 -1.0698857 1.813217
## 13 745046730 2023.609   102      105 81.42869 158.2504 -0.6831143 1.825103
## 14 745046730 2023.609   102      105 81.42867 158.2304 -0.7025393 1.837583
## 15 745046730 2023.609   102      105 81.42865 158.2104 -0.7252607 1.850657
## 16 745046730 2023.609   102      105 81.42863 158.1904 -0.7655750 1.864920
```

Exercise: Reading data

- Try and run the code from the previous slide
- Try and save output to a file
- Plot the location of the measurements
- Plot the elevation vs longitude
- If you are familiar with a GIS software you can try and import the data there
- **Hint** in R you can plot via the function `plot`

```
# To plot
plot(dat$lon,dat$lat,col='blue')
```

Part 2 Extracting data using a shapefile

The following example demonstrates how to extract along-track Sentinel-3 data inside a lake shapefile

```
myshape<- 'mask/Tajikistan_2_simple.shp' # path to shapefile
datin<-getInShape(dat,myshape,colid=2,doplot=FALSE)
```

Exercise: Extracting data via a shapefile

- Try and extract data inside the polygon lake mask
- Try and activate the plot function

Part 3 Generating lake level time series

For this part we will use Sentinel-3 data that is already extracted and contained in the file `lakedata_4610001882.csv`.

```
datafile<-'lakedata_4610001882.csv'
dat<-read.csv(datafile)

head(dat)
```

```
##     timesec     time cycle sattrack      lat      lon   height     geoid
## 1 513670162 2016.277     3       34 38.91159 64.61421 284.3958 -36.40480
## 2 516002963 2016.350     4       34 38.93669 64.63011 240.9670 -36.39347
## 3 516002963 2016.350     4       34 38.93382 64.62918 240.8956 -36.39619
## 4 516002963 2016.350     4       34 38.93094 64.62826 241.0265 -36.39886
## 5 516002963 2016.350     4       34 38.91944 64.62455 226.9350 -36.40924
## 6 516002963 2016.350     4       34 38.91657 64.62362 227.1632 -36.41170
##       lakeid
## 1 4610001882
## 2 4610001882
## 3 4610001882
## 4 4610001882
## 5 4610001882
## 6 4610001882
```

Exercise: plotting data

- Try and plot the lake height as a function of time
- Notice the outliers
- You can add the option `ylim` to get a better sense of the lake level variations.
- Hint use ?plot

We will now use the packages `tsHydro` to reconstruct the time series From `tsHydro` we will use the functions

- `get.TS`, reconstruct the time series via a simple state-space model
- `plot`, plot the reconstructed time series
- `export.tsHydro`, save model to a file
- `summary`, print model performance on the screen

```
#add track information to data needed in tsHydro
dat$track<-as.integer(as.factor(dat$time))
fit<-get.TS(dat) #reconstruct time series
plot(fit,addError=TRUE) #plot fitted model
# create an output file name similar to input file name
tsfile<-paste0('ts_',sub(".csv",".dat",datafile))
export.tsHydro(fit,tsfile) # export model to file
```

Exercise: Time series reconstruction

- Try and run the code from the previous slide
- Is the model affected by the outliers you observed in the data

- Try and use the function summary to see the model information

Summary of scripts

Putting it all together read and extract data

```
# The following script is available in the file 'readAndExtractS3.R'
source('S3Functions.R')
datadir<-'testdata'
#path to shape file
myshape<- '~/projects/WorldWater/data/Tajikistan/mask/Tajikistan_2_simple.shp'
#list files in the folder 'testdata'
myfiles<-dir(datadir,full.names=TRUE,recursive=TRUE,pattern='standard_measurement')
# The function doOneFileS3 will read and extract data for one S3 file see S3Functions.R
# 'lapply' loops over all files
mydat<-lapply(1:length(myfiles),function(i)doOneFileS3(myfiles[i],myshape))
mydat<-do.call(rbind,mydat)
Write.csv(mydat,file='mylake_data.csv',row.names=FALSE, quote=FALSE)
```

Putting it all together reconstruct time series

```
# The following script is available in the file 'reconstructTimeSeries.R'
source('S3Functions.R')
datafile<-'lakedata_4610001882.csv'
dat<-read.csv(datafile)
#add track information to data
dat$track<-as.integer(as.factor(dat$time))
fit<-get.TS(dat)
plot(fit,addError=TRUE)
tsfile<-paste0('ts_',sub(".csv",".dat",datafile))
export.tsHydro(fit,tsfile)
```