Using MODISTools (0.93.9)

Sean Tuck

2013-03-26

Contents

1 Introduction

The MODISTools R package is a set of tools for downloading and working with NASA's MODIS remotely-sensed data. The package retrieves data from the LP DAAC data archive, via their SOAP web service. Functions download data as a batch process, and save subsets in text files that can be returned to at a later date. Additional functions can provide summaries of this data and prepare the data to a format ready for application in R; if you have other data that you wish to relate MODIS data to, downloaded data can be appended to your original dataset. Other ancillary functions can help to get input arguments into the correct format.

This vignette provides a worked example for using MODISTools. A dataset of time-series – lat-long coordinates with start and end dates – to collect MODIS data for, will be used to show a complete workflow for how someone might use MODISTools. We will prepare input information for a subset request, download subsets of Enhanced Vegetation Index (EVI) and land cover data for the specified locations, and process these data to analyse land processes at these locations. Note that you will need an internet connection to run this worked example yourself, and that it will download files to your computer.

2 Format the data

We have some coordinates that we would like to extract MODIS data for. But the coordinates are not in the correct format. We need to make sure the coordinates we input for our subset request are in the WGS-1984 coordinate system, and are in decimal degrees format.

```
> data(ConvertExample)
> ConvertExample

             lat          long
1    51d24.106'N    0d38.018'W
2    51d24.922'N    0d38.772'W
3    51d24.106'N    0d38.664'W
4    51d24.772'N    0d38.043'W
5 51d24m51.106sN 0d38m56.018sW
6 51d24m37.922sN 0d38m31.772sW
7 51d24m42.106sN 0d38m17.664sW
8 51d24m47.772sN 0d38m42.043sW
```

These coordinates are WGS-1984 coordinates, but they are not in decimal degrees. We can use `ConvertToDD` to fix this.

```
> modis.subset <-
   ConvertToDD(XY = ConvertExample, LatColName = "lat", LongColName = "long")
> modis.subset <- data.frame(lat = modis.subset[ ,1], long = modis.subset[ ,2])
> modis.subset

       lat       long
1 51.40177 -0.6336333
2 51.41537 -0.6462000
3 51.40177 -0.6444000
4 51.41287 -0.6340500
5 51.41420 -0.6488939
6 51.41053 -0.6421589
7 51.41170 -0.6382400
8 51.41327 -0.6450119
```

What we also need to retrieve a time-series of MODIS data for these locations are dates. End dates for the time-series, and preferably start dates too. If we don't have start dates we can ask for a set number of years for each location instead. Let's retrieve data between 2003 and 2006. The dates can be specified as years or in POSIXlt date-time class (see `?POSIXlt`). In this case we can just use years.

```
> modis.subset$start.date <- rep(2003, nrow(modis.subset))
> modis.subset$end.date <- rep(2006, nrow(modis.subset))
```

That's all we need! Let's download our EVI data first.

3 Download the data

3.1 Specifying a subset request

The shortname code for the EVI product is "MOD13Q1". We can check the codes for all the products available using `GetProducts`, and we can find the shortname codes for all data bands within each product using `GetBands`.

```
> GetProducts()
```

```
 [1] "MCD12Q1"    "MCD12Q2"    "MCD43A1"    "MCD43A2"    "MCD43A4"    "MOD09A1"
 [7] "MOD11A2"    "MOD13Q1"    "MOD15A2"    "MOD15A2GFS" "MOD16A2"    "MOD17A2_51"
[13] "MOD17A3"    "MYD09A1"    "MYD11A2"    "MYD13Q1"    "MYD15A2"
```

```
> GetBands(Product = "MOD13Q1")
```

```
 [1] "250m_16_days_blue_reflectance"          "250m_16_days_MIR_reflectance"
 [3] "250m_16_days_NIR_reflectance"           "250m_16_days_pixel_reliability"
 [5] "250m_16_days_red_reflectance"           "250m_16_days_relative_azimuth_angle"
 [7] "250m_16_days_sun_zenith_angle"          "250m_16_days_view_zenith_angle"
 [9] "250m_16_days_VI_Quality"                "250m_16_days_NDVI"
[11] "250m_16_days_EVI"                       "250m_16_days_composite_day_of_the_year"
```

We will download EVI data at 250m pixel resolution, which is available at 16-day intervals. The shortname code for this data band is 250m_16_days_EVI. We will collect quality control data for these pixels too, which is available from the 250m_16_days_pixel_reliability band (and 250m_16_days_VI_Quality too).

We can check that the time-series of MODIS data we want is available for this data product by retrieving the dates for all available time-steps.

```
> GetDates(Product = "MOD13Q1", Lat = modis.subset$lat[1], Long = modis.subset$long[1])
```

The time-period available for the Vegetation Indices product covers 2003-2006 (the maximum shown is at the time this vignette was built), so we can proceed. When we download we also need to decide how large we want the tiles of data for each location to be. We specify this by entering the distance (km) above and below in each direction away from the central pixel, where the input coordinate is located, and then doing the same for left and right. The input must be whole km (integers) for each direction. As an example, if we specify `Size=c(1,1)` for this EVI data at 250m pixel resolution, it will retrieve a 9x9 pixel tile for each location, centred on the input coordinate. The tiles this size will be downloaded at the locations for each time-step that falls between the start and end dates. `Size=c(0,0)` would specify only the central pixel. The maximum size tile surrounding a location is `Size=c(100,100)`.

3.2 MODISSubsets

The download will write the MODIS data to ASCII files for each location subset specified. We can specify the directory that we would like to save downloaded files in, using the `SaveDir` argument below. In the code below, downloaded files will be written to your working directory; if you would prefer the files to be written elsewhere change `SaveDir`. But we will access these files later, so remember to request the files from the same directory.

```
> MODISSubsets(LoadDat = modis.subset, Products = "MOD13Q1",
               Bands = c("250m_16_days_EVI", "250m_16_days_pixel_reliability"),
               Size = c(1,1))
```

Each ASCII file is a different subset location. In each ASCII file, each row is a different time-step in the time-series. If multiple data bands have been downloaded for this subset, they will all be contained in the same ASCII file for that subset.

Here is an example of the strings of data that are downloaded for pixels at each time-step and data band:

```
> subset.string <- read.csv(paste(list.files(pattern = ".asc")[1],
                                  header = FALSE, as.is = TRUE))
> subset.string[1, ]
```

```
                                                    V1      V2      V3
1 MOD13Q1.A2004001.h17v03.005.2007234110215.250m_16_days_EVI MOD13Q1 A2004001
                                      V4            V5   V6   V7   V8   V9  V10  V11
1 Lat51.41327Lon-0.645011944444444Samp9Line9 2.007234e+12 2627 2627 4135 4119 4123 2986
   V12  V13  V14  V15  V16  V17  V18  V19  V20  V21  V22  V23  V24  V25  V26  V27  V28
1 3124 2449 2449 2639 2258 2258 2356 2356 2986 2986 2336 2700 2639 2248 2248 2131 2463
   V29  V30  V31  V32  V33  V34  V35  V36  V37  V38  V39  V40  V41  V42  V43  V44  V45
1 2463 2834 2834 2700 2824 2492 2139 2131 2310 2310 2300 2834 2522 2498 2492 2139 2139
   V46  V47  V48  V49  V50  V51  V52  V53  V54  V55  V56  V57  V58  V59  V60  V61  V62
1 2092 2318 2300 2323 2323 2498 2498 2146 2172 2092 2318 2318 2437 2323 2723 2523 2523
   V63  V64  V65  V66  V67  V68  V69  V70  V71  V72  V73  V74  V75  V76  V77  V78  V79
1 2172 2172 2317 2317 2437 2634 2444 1988 2523 2207 2378 2378 2292 2292 2820 2589 1988
   V80  V81  V82  V83  V84  V85  V86
1 1962 1962 2237 2378 2378 2381 2727
```

A download log file will also be written, displaying all the unique subsets found in the dataset, and confirmation of download success for each.

3.3 MODISTransects

Alternatively, we may want transects of MODIS data. This is easily done by specifying start and end points for transects and calling `MODISTransects`. Our data here does not have coordinates that specify transect end points yet, so we need to calculate them using `EndCoordinates`.

```
> names(modis.subset) <- c("start.lat", "start.long", "start.date", "end.date")
> EndCoordinates(LoadDat = modis.subset, Distance = 1000, Angle = 60,
                 AngleUnits = "degrees")
> modis.transect <- read.csv(list.files(pattern = "Transect Coordinates"))
> MODISTransects(LoadData = modis.transect, Product = "MOD13Q1",
     Bands = c("250m_16_days_EVI", "250m_16_days_pixel_reliability"),
     Size = c(0,0), StartDate = TRUE)
```

4 Process the data

4.1 MODISSummaries

Now we have downloaded the EVI data, we can find average each pixel over time, to produce one tile of mean EVI pixels at each subset location. We can use `MODISSummaries` for this. The function will also take this processed data and append it to your original files containing all the subset information (`modis.subset`). This will write two files to the specified directory. We downloaded quality control data for each pixel alongside our EVI data, so `MODISSummaries` can also check for poor quality and missing data. These data will be removed and replaced with `NA`s. The threshold for defining what is good and poor quality is set by the user: the scores for highest quality is 0, and the score for lowest quality is 3 or 5, depending on the data band. To see how quality control information is defined for each data type, go to the **MODIS Products Table**. We need to specify the range of valid data for EVI, the value that denotes missing data, and the scale factor that is applied to the data, which are all available from the same web page.

```
> MODISSummaries(LoadDat = modis.subset, Product = "MOD13Q1", Bands = "250m_16_days_EVI",
                 ValidRange = c(-2000,10000), NoDataFill = -3000, ScaleFactor = 0.0001,
```

```
                QualityScreen = TRUE, QualityBand = "250m_16_days_pixel_reliability",
                QualityThreshold = 0)
```

If you want to screen data for quality without all the other things that `MODISSummaries` does, you can call the more general `QualityCheck`, which is an internal function for `MODISSummaries`.

4.2 ExtractTile

Also, if large subset tiles are downloaded for each location, there may be times when we want to extract a smaller tile from within this subset, rather than downloading again to retrieve the nested data we want. This can be done using `ExtractTile`. We will use the file just written from our call to `MODISSummaries`, retrieve the smaller subset we want, and arrange them into tiles to compare before and after.

```
> TileExample <- read.csv(list.files(pattern = "MODIS_Data"))
> TileExample <- TileExample[ ,which(grepl("band.pixels", names(TileExample)))]
```

Pixels in a tile are on the same row. See that using `ExtractTile` takes away some of the columns.

```
> dim(TileExample)

[1]  8 81

> dim(ExtractTile(Data = TileExample, Rows = c(9,2), Cols = c(9,2), Grid = FALSE))

[1]  8 25

> head(ExtractTile(Data = TileExample, Rows = c(9,2), Cols = c(9,2), Grid = FALSE),
      n = 2)

          [,1]      [,2]     [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
[1,] 0.3507588 0.3778630 0.401367 0.3664756 0.3114811 0.3660027 0.4076705 0.3843886
[2,] 0.3654468 0.3644941 0.382880 0.3970886 0.4155620 0.4385893 0.3623270 0.3677757
          [,9]     [,10]     [,11]     [,12]     [,13]     [,14]     [,15]     [,16]
[1,] 0.3671898 0.3770149 0.3846099 0.4011764 0.3454266 0.3313671 0.3954960 0.3863663
[2,] 0.3840878 0.3954889 0.3991618 0.3288403 0.3849842 0.3895216 0.3875111 0.3976902
         [,17]     [,18]     [,19]     [,20]     [,21]     [,22]     [,23]     [,24]
[1,] 0.4266230 0.3572599 0.3609461 0.4206966 0.4583811 0.4923791 0.4212212 0.4200632
[2,] 0.3680142 0.3887727 0.4036611 0.3817324 0.4083904 0.4237785 0.4024520 0.3819636
         [,25]
[1,] 0.4498460
[2,] 0.3837451
```

We can look at the first subset and arrange the pixels into a tile to visually show what `ExtractTile` has done.

```
> matrix(TileExample[1, ], nrow = 9, ncol = 9, byrow = TRUE)

      [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
 [1,] 0.4035009 0.368406  0.3369911 0.3706116 0.405156  0.4077885 0.380869  0.3764521
 [2,] 0.3779488 0.3695678 0.329654  0.3499113 0.3987212 0.394887  0.4079352 0.4933911
 [3,] 0.3573749 0.3605016 0.3507588 0.3660027 0.3846099 0.3863663 0.4583811 0.5585599
```

```
 [4,] 0.3533674 0.3489834 0.377863  0.4076705 0.4011764 0.426623  0.4923791 0.5042675
 [5,] 0.3471202 0.3467685 0.401367  0.3843886 0.3454266 0.3572599 0.4212212 0.4439073
 [6,] 0.3445654 0.32862   0.3664756 0.3671898 0.3313671 0.3609461 0.4200632 0.4045281
 [7,] 0.3914093 0.322921  0.3114811 0.3770149 0.395496  0.4206966 0.449846  0.4017315
 [8,] 0.4221513 0.367898  0.3645603 0.4070896 0.3875095 0.3733892 0.3743215 0.3597581
 [9,] 0.3705793 0.3538647 0.3312581 0.3011145 0.2922328 0.2747998 0.2930625 0.3217153
      [,9]
 [1,] 0.3499755
 [2,] 0.4769236
 [3,] 0.4956572
 [4,] 0.4170771
 [5,] 0.4335761
 [6,] 0.3833537
 [7,] 0.371729
 [8,] 0.3566241
 [9,] 0.3520446
```

```
> ExtractTile(Data = TileExample, Rows = c(9,2), Cols = c(9,2), Grid = TRUE)[ , ,1]
```

```
          [,1]      [,2]      [,3]      [,4]      [,5]
[1,] 0.3507588 0.3660027 0.3846099 0.3863663 0.4583811
[2,] 0.3778630 0.4076705 0.4011764 0.4266230 0.4923791
[3,] 0.4013670 0.3843886 0.3454266 0.3572599 0.4212212
[4,] 0.3664756 0.3671898 0.3313671 0.3609461 0.4200632
[5,] 0.3114811 0.3770149 0.3954960 0.4206966 0.4498460
```

Arrangement of the pixels into tiles this way can be optionally set with a call to `ExtractTile`. The order for the strings of pixel data in the downloaded ASCII files is by row, so `matrix(..., byrow=TRUE)` can arrange the pixels correctly (see above).

4.3 LandCover

Let's do the same as above but download data on land cover classes for the same subsets.

```
> MODISSubsets(LoadDat = modis.subset, Product = "MCD12Q1", Bands = "Land_Cover_Type_1",
             Size = c(1,1))
```

We can use `LandCover` to retrieve some summaries of land cover in each tile. This will tell us the most common land cover type, the total number of distinct land cover types, and Simpson's D and evenness measures to express landscape diversity and heterogeneity in these tiles. Let's retrieve these summaries from the land cover subset files we just downloaded.

```
> LandCover(Band = "Land_Cover_Type_1")
> land.summary <- read.csv(list.files(pattern = "MODIS_Land_Cover_Summary"))
> head(land.summary)
```

```
       lat       long       date        modis.band                 most.common richness
1 51.40177 -0.6336333 2004-01-01 Land_Cover_Type_1 Deciduous Needleleaf forest        3
2 51.40177 -0.6336333 2005-01-01 Land_Cover_Type_1 Evergreen Needleleaf forest        3
3 51.40177 -0.6336333 2006-01-01 Land_Cover_Type_1 Deciduous Needleleaf forest        3
```

```
4 51.40177 -0.6444000 2004-01-01 Land_Cover_Type_1  Evergreen Broadleaf forest        3
5 51.40177 -0.6444000 2005-01-01 Land_Cover_Type_1 Deciduous Needleleaf forest        4
6 51.40177 -0.6444000 2006-01-01 Land_Cover_Type_1 Deciduous Needleleaf forest        4
  simpsons.d simpsons.evenness no.data.fill
1   2.880184         0.9600614    0% (0/25)
2   2.880184         0.9600614    0% (0/25)
3   2.880184         0.9600614    0% (0/25)
4   2.729258         0.9097525    0% (0/25)
5   2.777778         0.6944444    0% (0/25)
6   2.659574         0.6648936    0% (0/25)
```