Acessing time series information in SITS

Rolf Simoes *National Institute for Space Research (INPE), Brazil*
Gilberto Camara *National Institute for Space Research (INPE), Brazil*
Pedro R. Andrade *National Institute for Space Research (INPE), Brazil*

This vignette describes how to access information from time series in SITS.

Introduction

This vignette complements the main sits vignette and provides additional information about handling time series data in SITS.

Data structures for satellite time series

The `sits` package requires a set of time series data, describing properties in spatio-temporal locations of interest. For land use classification, this set consists of samples provided by experts that take *in-situ* field observations or recognize land classes using high resolution images. The package can also be used for any type of classification, provided that the timeline and bands of the time series (used for training) match that of the data cubes.

For handling time series, the package uses a `sits tibble` to organize time series data with associated spatial information. A `tibble` is a generalization of a `data.frame`, the usual way in *R* to organise data in tables. Tibbles are part of the `tidyverse`, a collection of R packages designed to work together in data manipulation [Wickham and Grolemund, 2017]. As a example of how the `sits` tibble works, the following code shows the first three lines of a tibble containing 2,115 labelled samples of land cover in Mato Grosso state of Brazil. It is the most important agricultural frontier of Brazil and it is the largest producer of soybeans, corn, and cotton. The samples contain time series extracted from the MODIS MOD13Q1 product from 2000 to 2016, provided every 16 days at 250-meter spatial resolution in the Sinusoidal projection. Based on ground surveys and high resolution imagery, it includes 425 samples of nine classes: “Forest”, “Cerrado”, “Pasture”, “Soybean-fallow”, “Fallow-Cotton”, “Soybean-Cotton”, “Soybean-Corn”, “Soybean-Millet”, and “Soybean-Sunflower”.

```
# data set of samples
data(samples_mt_4bands)
samples_mt_4bands[1:3,]
```

```
## # A tibble: 3 x 9
##   id_sample cluster_label longitude latitude start_date end_date   label cube
```

```
##       <int> <chr>            <dbl>    <dbl> <date>     <date>     <chr> <chr>
## 1       1 Pasture         -55.2   -10.8  2013-09-14 2014-08-29 Past~ MOD1~
## 2       2 Pasture         -57.8    -9.76 2006-09-14 2007-08-29 Past~ MOD1~
## 3       3 Pasture         -51.9   -13.4  2014-09-14 2015-08-29 Past~ MOD1~
## # ... with 1 more variable: time_series <list>
```

A sits tibble contains data and metadata. The first six columns contain the metadata: spatial and temporal information, label assigned to the sample, and the data cube from where the data has been extracted. The spatial location is given in longitude and latitude coordinates for the "WGS84" ellipsoid. For example, the first sample has been labelled "Cerrado, at location (−58.5631, −13.8844), and is considered valid for the period (2007-09-14, 2008-08-28). Informing the dates where the label is valid is crucial for correct classification. In this case, the researchers involved in labeling the samples chose to use the agricultural calendar in Brazil, where the spring crop is planted in the months of September and October, and the autumn crop is planted in the months of February and March. For other applications and other countries, the relevant dates will most likely be different from those used in the example. The time_series column contains the time series data for each spatiotemporal location. This data is also organized as a tibble, with a column with the dates and the other columns with the values for each spectral band.

Utilities for handling time series

The sits package provides functions for data manipulation and displaying information for sits tibbles. For example, sits_labels() shows the labels of the sample set and their frequencies.

```r
sits_labels(samples_mt_4bands)
```

```
## # A tibble: 9 x 3
##   label          count   prop
##   <chr>          <int>  <dbl>
## 1 Cerrado          379 0.200
## 2 Fallow_Cotton     29 0.0153
## 3 Forest           131 0.0692
## 4 Pasture          344 0.182
## 5 Soy_Corn         364 0.192
## 6 Soy_Cotton       352 0.186
## 7 Soy_Fallow        87 0.0460
## 8 Soy_Millet       180 0.0951
## 9 Soy_Sunflower     26 0.0137
```

In many cases, it is useful to relabel the data set. For example, there may be situations when one wants to use a smaller set of labels, since samples in one label on the original set may not be distinguishable from samples with other labels. We then could use sits_relabel(), which requires a conversion list (for details, see ?sits_relabel).

Given that we have used the tibble data format for the metadata and and the embedded time series, one can use the functions from `dplyr`, `tidyr` and `purrr` packages of the `tidyverse` [Wickham and Grolemund, 2017] to process the data. For example, the following code uses `sits_select_bands()` to get a subset of the sample data set with two bands (NDVI and EVI) and then uses the `dplyr::filter()` to select the samples labelled either as "Cerrado" or "Pasture".

```
# select NDVI band
samples_ndvi.tb <- sits_select_bands(samples_mt_4bands, ndvi)
# select only samples with Cerrado label
samples_cerrado.tb <-
    dplyr::filter(samples_ndvi.tb, label == "Cerrado")
```

Time series visualisation

Given a small number of samples to display, `plot` tries to group as many spatial locations together. In the following example, the first 15 samples of "Cerrado" class refer to the same spatial location in consecutive time periods. For this reason, these samples are plotted together.

```
# plot the first sample
plot(samples_cerrado.tb[1:15,])
```

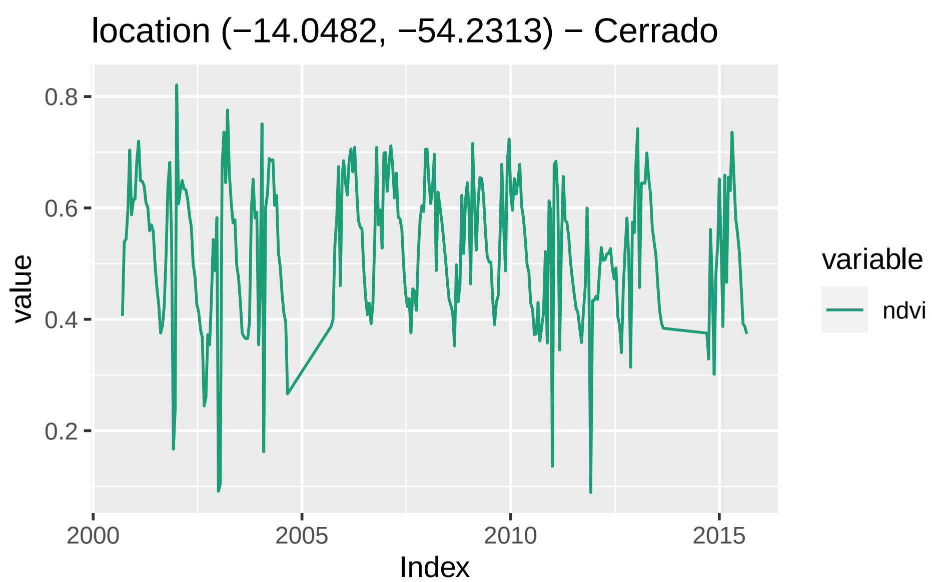

Figure 1: Plot of the first 'Cerrado' sample from data set

For a large number of samples, where the amount of individual plots would be substantial, the default visualization combines all samples together in a single temporal interval (even if they belong to different years). All samples with the same band and label are aligned to a common time interval. This plot is useful to show the spread of

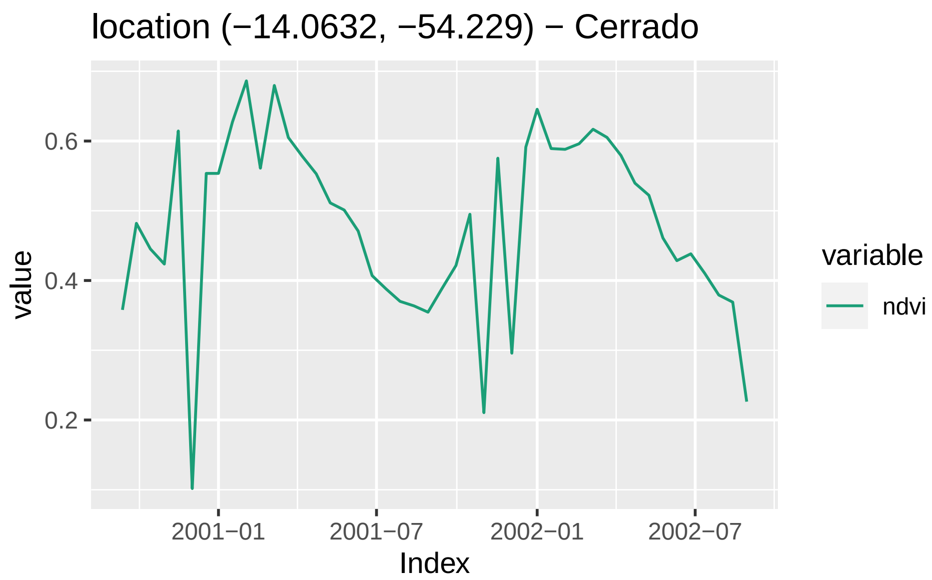

Figure 2: Plot of the first 'Cerrado' sample from data set

values for the time series of each band. The strong red line in the plot shows the median of the values, while the two orange lines are the first and third interquartile ranges. The documentation of `plot.sits()` has more details about the different ways it can display data.

```
# plot all cerrado samples together
plot(samples_cerrado.tb)
```

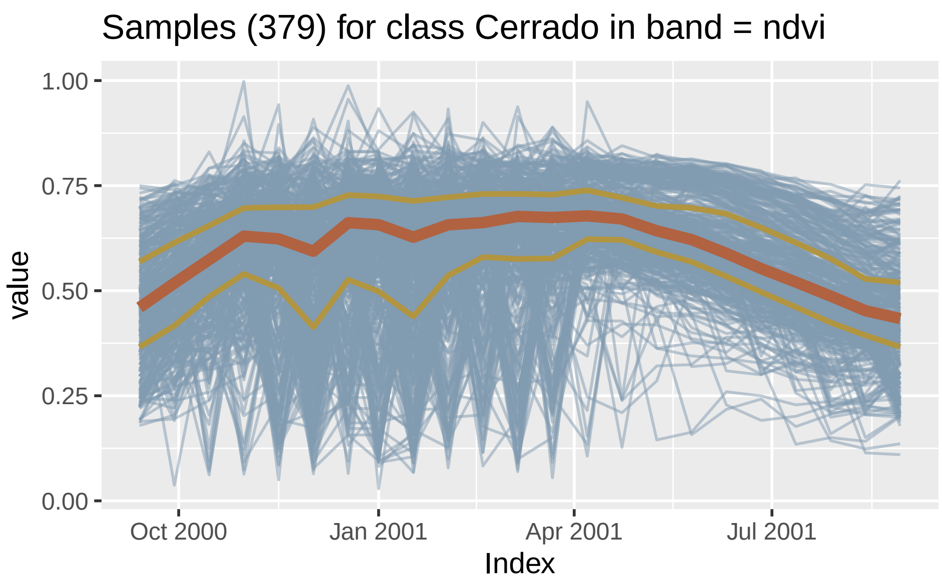

Figure 3: Plot of all Cerrado samples from data set

Obtaining time series data

To get a time series in SITS, one has to create a data cube first, as described above. Alternatively, the time series can also be converted from data stored in the ZOO format [Zeileis and Grothendieck, 2005]. Users can request one or more time series points from a data cube by using `sits_get_data()`. This function provides a general means of access to image time series. Given data cue, the user provides the latitude and longitude of the desired location, the bands, and the start date and end date of the time series. If the start and end dates are not provided, it retrieves all the available period. The result is a tibble that can be visualized using `plot()`.

```
# define the data cube "MOD13Q1" using the WTSS service
# In this case, the WTSS service is run by a server in INPE Brazil
wtss_cube <- sits_cube(type = "WTSS",
                       URL = "http://www.esensing.dpi.inpe.br/wtss",
                       name = "MOD13Q1")
# a point in the transition forest to pasture in Northern MT
# obtain a time series from the WTSS server for this point
series.tb <- sits_get_data(cube      = wtss_cube,
                           longitude = -55.57320,
                           latitude  = -11.50566,
                           bands     = c("ndvi", "evi"))
plot(series.tb)
```

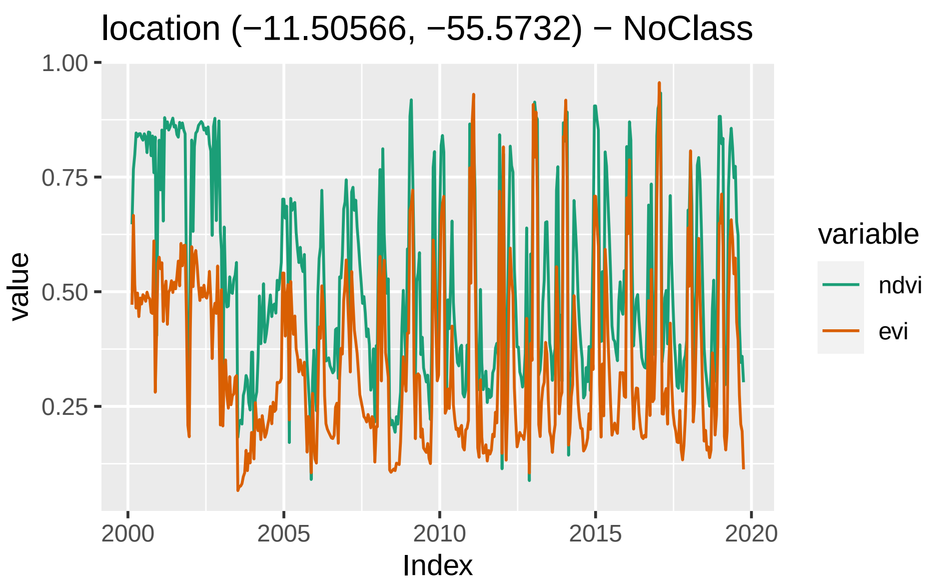

Figure 4: NDVI and EVI time series fetched from WTSS service.

A useful case is when a set of labelled samples are available to be used as a training data set. In this case, one usually has trusted observations which are labelled and commonly stored in plain text CSV files. Function `sits_get_data()` can get a CSV file path as an argument. The CSV file must provide, for each time series, its latitude and longitude, the start and end dates, and a label associated to a ground sample. An example of a CSV file used is shown below:

```
# print the first line of a CSV file used to retrieve data
csv_sinop <- system.file("extdata/samples/samples_matogrosso.csv", package = "sits")
head(read.csv(csv_sinop))
```

```
## # A tibble: 6 x 6
##      id longitude latitude start_date end_date   label
##   <int>     <dbl>    <dbl> <chr>      <chr>      <chr>
## 1     1     -55.0    -15.2 2015-09-14 2016-08-28 Pasture
## 2     2     -55.0    -15.2 2015-09-14 2016-08-28 Pasture
## 3     3     -55.0    -15.2 2015-09-14 2016-08-28 Pasture
## 4     4     -46.6    -10.4 2004-09-13 2005-08-29 Cerrado
## 5     5     -46.4    -10.9 2007-09-13 2008-08-29 Cerrado
## 6     6     -46.4    -10.9 2006-09-13 2007-08-29 Cerrado
```

```
# read the first three samples from the CSV file
csv_data <- sits_get_data(cube = wtss_cube, file = csv_sinop, .n_max_csv = 3)
```

```
## All points have been retrieved from WTSS service
```

```
csv_data
```

```
## # A tibble: 3 x 7
##   longitude latitude start_date end_date   label   cube    time_series
##       <dbl>    <dbl> <date>     <date>     <chr>   <chr>   <list>
## 1     -55.0    -15.2 2015-09-14 2016-08-28 Pasture MOD13Q1 <tibble [23 x 7]>
## 2     -55.0    -15.2 2015-09-14 2016-08-28 Pasture MOD13Q1 <tibble [23 x 7]>
## 3     -55.0    -15.2 2015-09-14 2016-08-28 Pasture MOD13Q1 <tibble [23 x 7]>
```

A common situation is when users have samples available as shapefiles in point format. Since shapefiles contain only geometries, we need to provide information about the start and end times for which each label is valid. in this case, one should use the function `sits_get_data()` to retrieve data from a data cube based on the contents of the shapefile. The parameter `shp_attr` (optional) indicates the name of the column on the shapefile which contains the label to be associated to each time series; the parameter `.n_shp_pol` (defaults to 20) determines the number of samples to be extracted from each polygon.

```
# define the input shapefile (consisting of POLYGONS)
shp_file <- system.file("extdata/shapefiles/parcel_agriculture.shp",
                        package = "sits")
# set the start and end dates
```

```r
start_date <- lubridate::ymd("2012-08-29")
end_date   <- lubridate::ymd("2013-08-13")
# define the name of attribute of the shapefile that contains the label
shp_attr <- "ext_na"
# define the number of samples to extract from each polygon
.n_shp_pol <- 10
# read the points in the shapefile and produce a CSV file
data <- sits_get_data(cube = wtss_cube,
                      file = shp_file,
                      start_date = start_date,
                      end_date = end_date,
                      shp_attr = shp_attr,
                      .n_shp_pol = .n_shp_pol)
data
```

```
## # A tibble: 10 x 7
##    longitude latitude start_date end_date   label       cube   time_series
##        <dbl>    <dbl> <date>     <date>     <chr>       <chr>  <list>
## 1     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 2     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 3     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 4     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 5     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 6     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 7     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 8     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 9     -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
## 10    -55.2    -15.4 2012-08-29 2013-08-13 Soja_Algodao MOD13~ <tibble [22 x 7~
```

The SITS package enables uses to create data cube based on files. In this case, these files should be organized as `raster bricks`. A RasterBrick is a multi-layer raster object used by the *R* `raster` package. Each brick is a multi-layer file, containing different time instances of one spectral band. To allow users to create data cubes based on files, SITS needs to know what is the timeline of the data sets and what are the names of the files that contain the RasterBricks. The example below shows one bricks containing 392 time instances of the "ndvi" band for the years 2000 to 2016. The timeline is available as part of the SITS package. In this example, as in most cases using raster bricks, images are stored as GeoTiff files.

Since GeoTiff files do not contain information about satellites and sensors, it is best practice to provide information on satellite and sensor.

```r
# Obtain a raster brick with 23 instances for one year
# Select the band "ndvi", "evi" from bricks  available in the "sits" package
ndvi_file <- system.file("extdata/raster/mod13q1/sinop-crop-ndvi.tif",
                         package = "sits")
```

```r
# Obtain the associated timeline
timeline <- data("timeline_2000_2017")

# create a raster metadata file based on the information about the files
raster_cube <- sits_cube(type = "BRICK",
                         name = "Sinop",
                         satellite = "TERRA",
                         sensor = "MODIS",
                         timeline = timeline,
                         bands = c("ndvi"),
                         files = c(ndvi_file))
```

```
## Warning: All formats failed to parse. No formats found.
```

```r
# get information on the data cube
raster_cube %>% dplyr::select(type, URL, satellite, sensor)
```

```
## # A tibble: 1 x 4
##   type  URL              satellite sensor
##   <chr> <chr>            <chr>     <chr>
## 1 BRICK http://127.0.0.1 TERRA     MODIS
```

```r
# get information on the coverage
raster_cube %>% dplyr::select(xmin, xmax, ymin, ymax)
```

```
## # A tibble: 1 x 4
##         xmin      xmax      ymin      ymax
##        <dbl>     <dbl>     <dbl>     <dbl>
## 1 -6054361. -6051117. -1282895. -1280347.
```

To create the raster cube, we a set of consistent raster bricks (one for each satellite band) and a `timeline` that matches the input images of the raster brick. Once created, the coverage can be used either to retrieve time series data from the raster bricks using `sits_get_data()` or to do the raster classification by calling the function `sits_classify`.

References

Hadley Wickham and Garrett Grolemund. *R for Data Science: Import, Tidy, Transform, Visualize, and Model Data.* O'Reilly Media, Inc., 2017.

Achim Zeileis and Gabor Grothendieck. zoo: S3 infrastructure for regular and irregular time series. *Journal of Statistical Software*, 14(6):1–27, 2005.