qgs: Handling QGIS Project Files

Pedro R. Andrade

January 23, 2018

Abstract

This package provides a simple solution to the data interface between QGIS and R. It allows users to handle QGIS project files directly, simplifying the way one can read geospatial data into R.

Introduction

This package allows users to handle QGIS project files, with extension `.qgs`. These files store a set of layers, pointing each of them to a data source, like a shapefile, a tiff, or a WFS (Web Feature Service). Given a project created in QGIS, it is possible to read data directly into R using only the name of the layers, without worrying about where the data is stored and which R packages, functions, and parameters are necessary to read the data. It is also possible to generate the source code to read the data directly from the data source.

Project and Layer

Let us start by defining where the qgs file is located as an R variable. In this tutorial, I will use a file available within qgs package.

```r
file = system.file("geo/amazonia.qgs", package = "qgs")
```

The function to open a given QGIS project is called `openProject()`. It gets as argument a string with the file to be opened and returns an object of type `qgsProject`.

```r
proj = openProject(file)
```

```
## The following variables were created: 'indigenous', 'limit', 'ports', 'prodes', 'roads'
```

When the project is opened, `openProject()` also declares all the layers of the project as R objects if there are no objects with such names in the global environment. In the code above, five variables were declared. If you print the variable storing the project, it shows the available layers again:

```r
proj
```

```
## An object of class qgsProject (QGIS Project)
## File: "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/qgs/geo/amazonia.qgs"
## Layers: "indigenous", "limit", "ports", "prodes", "roads"
```

Each declared layer belongs to class `qgsLayer`. Two of them are shown below:

```r
limit
```

```
## An object of class qgsLayer (QGIS Layer)
## Name: "limit"
## Provider: "ogr"
## Source: "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/qgs/geo/amazonia-limit.shp"
```

```r
prodes
```

```
## An object of class qgsLayer (QGIS Layer)
## Name: "prodes"
## Provider: "gdal"
## Source: "/Library/Frameworks/R.framework/Versions/3.4/Resources/library/qgs/geo/amazonia-prodes.tif"
```

If the layers already exist as variables, they will not be declared. For example, if we open the same project again, we get the following output:

```
proj = openProject(file)
```

```
## Warning: Variables 'indigenous', 'limit', 'ports', 'prodes', 'roads'
## already exist and will not be replaced. Set replace = TRUE to overwrite
## them.
```

The argument `replace` allows objects to be replaced. In this case, no warning will be shown.

```
proj = openProject(file, replace = TRUE)
```

```
## The following variables were created: 'indigenous', 'limit', 'ports', 'prodes', 'roads'
```

Reading data

It is possible to read the data from a layer using `readData`. Note that this is the only function to read data of the package. It recognizes the data source and uses the proper function internally to read the data.

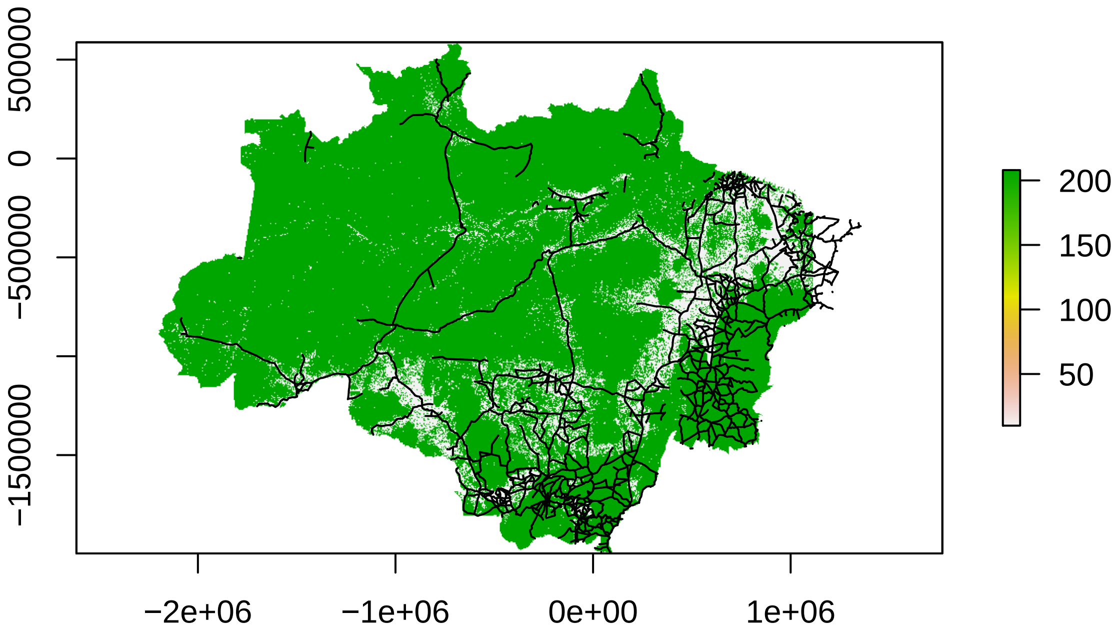
```
data = readData(prodes)
class(data)
```

```
## [1] "RasterLayer"
## attr(,"package")
## [1] "raster"
```

```
lines = readData(roads)
class(lines)
```

```
## [1] "SpatialLinesDataFrame"
## attr(,"package")
## [1] "sp"
```

```
plot(data)
plot(lines, add = T)
```



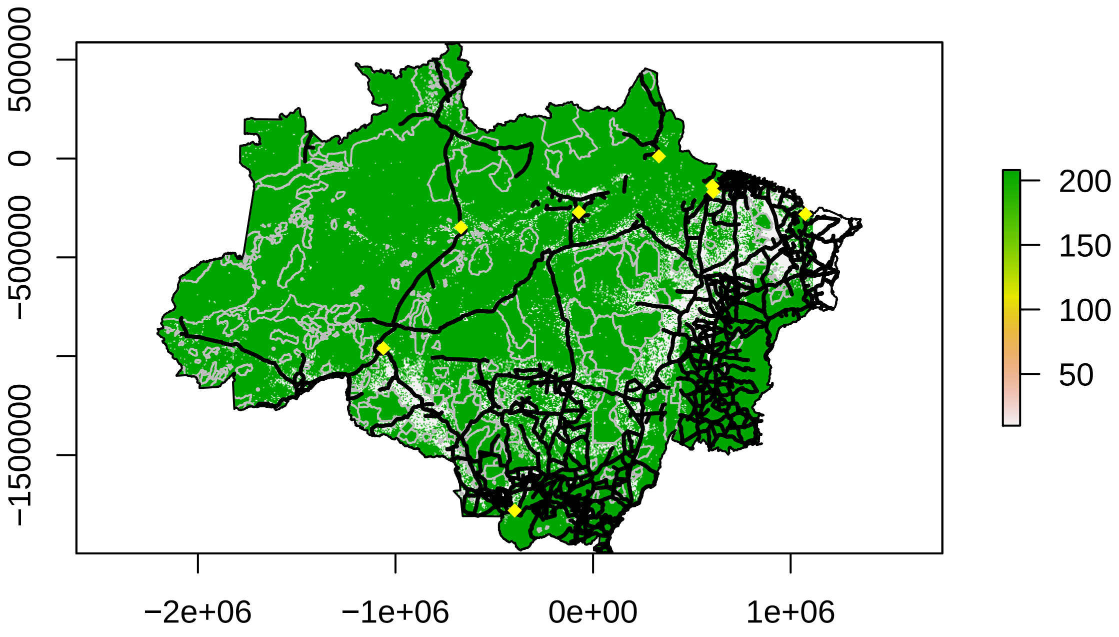
If one wants to know how to read the data from the data source of the layer without using the project file, it is possible to use `readDataCode()` that gets the layer as argument in the same way of `readData()`. This function generates code using the complete path for files (QGIS sometimes stores relative paths, which work properly only if the current directory of R is the same directory of the qgs file). This function is also useful when one wants to set additional arguments to the function call to read the data in a proper way.

```
readDataCode(prodes)
```

```
## require(raster)
## prodes = raster("/Library/Frameworks/R.framework/Versions/3.4/Resources/library/qgs/geo/amazonia-pro
```

Finally, it is also possible to plot the data directly from the layer, such as in the example below:

```
plot(prodes)
plot(indigenous, border = "gray", add = T)
plot(roads, lwd = 2, add = T)
plot(limit, add = T)
plot(ports, pch=18, col="yellow", add = T)
```



Another project

The last example uses WFS data stored in a project. In the same way of the first example, this one uses a project stored in package qgs.

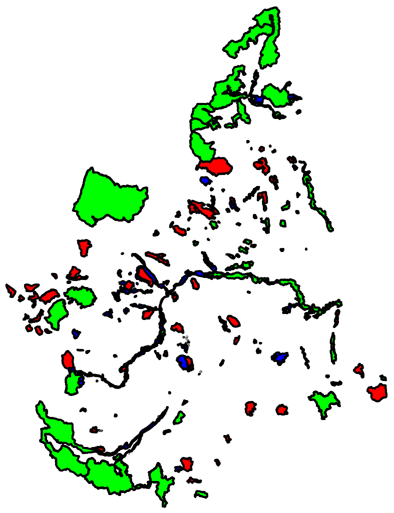
```
file = system.file("geo/piemonte.qgs", package = "qgs")

proj = openProject(file, replace = TRUE)
```

```
## The following variables were created: 'AreeProtette', 'SIC', 'ZPS'
```

```
# the code below might take some time as it downloads data from a WFS
areeprotette = readData(AreeProtette)
sic = readData(SIC)
zps = readData(ZPS)
```

The following code plots all the layers available within piemonte project.

```
plot(areeprotette, col = "blue")
plot(sic, col = "red", add = T)
plot(zps, col = "green", add = T)
```



To read data from a WMS, it is necessary to use `readOGR()`, as we can see using `readDataCode()`.

```
readDataCode(SIC)
```

```
## require(rgdal)
## SIC = readOGR(dsn = "WFS:http://geomap.reteunitaria.piemonte.it/ws/gsareprot/rp-01/areeprotwfs/wfs_g
```

Final remarks

More is still to be implemented. Some of future functionalities include:

- Adding new layers to qgs files from the R side;
- Allow reading only attributes (without geometry) whenever possible;
- Handling colors and styles;
- Finding and updating the recent qgs projects visible in QGIS.

If you have suggestions please visit the GitHup page.