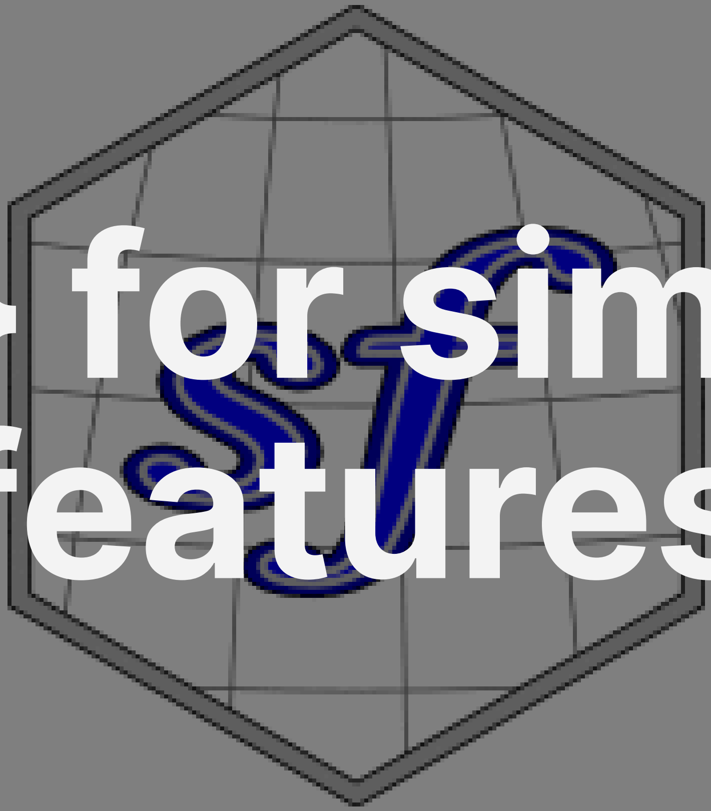
Getting map data into R

{sf} for simple features

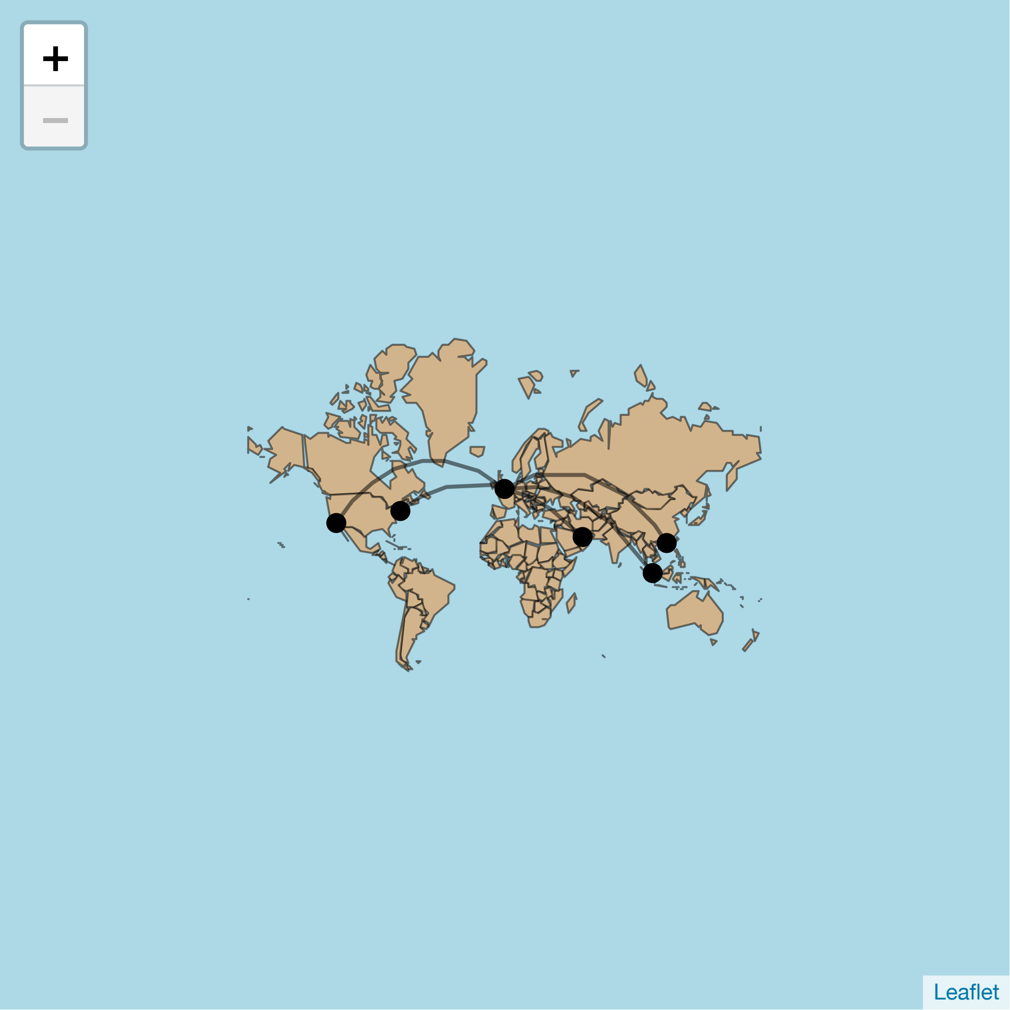


Most maps contain simple features

This map shows the **5 busiest passenger routes** in 2015
(by total seat kilometres).

It contains three types of feature:

- LINESTRING for the flight routes
- POINT for the airport location
- POLYGON for the country borders



{sf} implements simple features for R

- Simple features is a formal standard for organising and processing GIS data.
- The `{sf}` package fully implements the simple features standards in R
- The `{sf}` package ultimately provides us with a `data.frame()` like object that we can manipulate with the tidyverse.

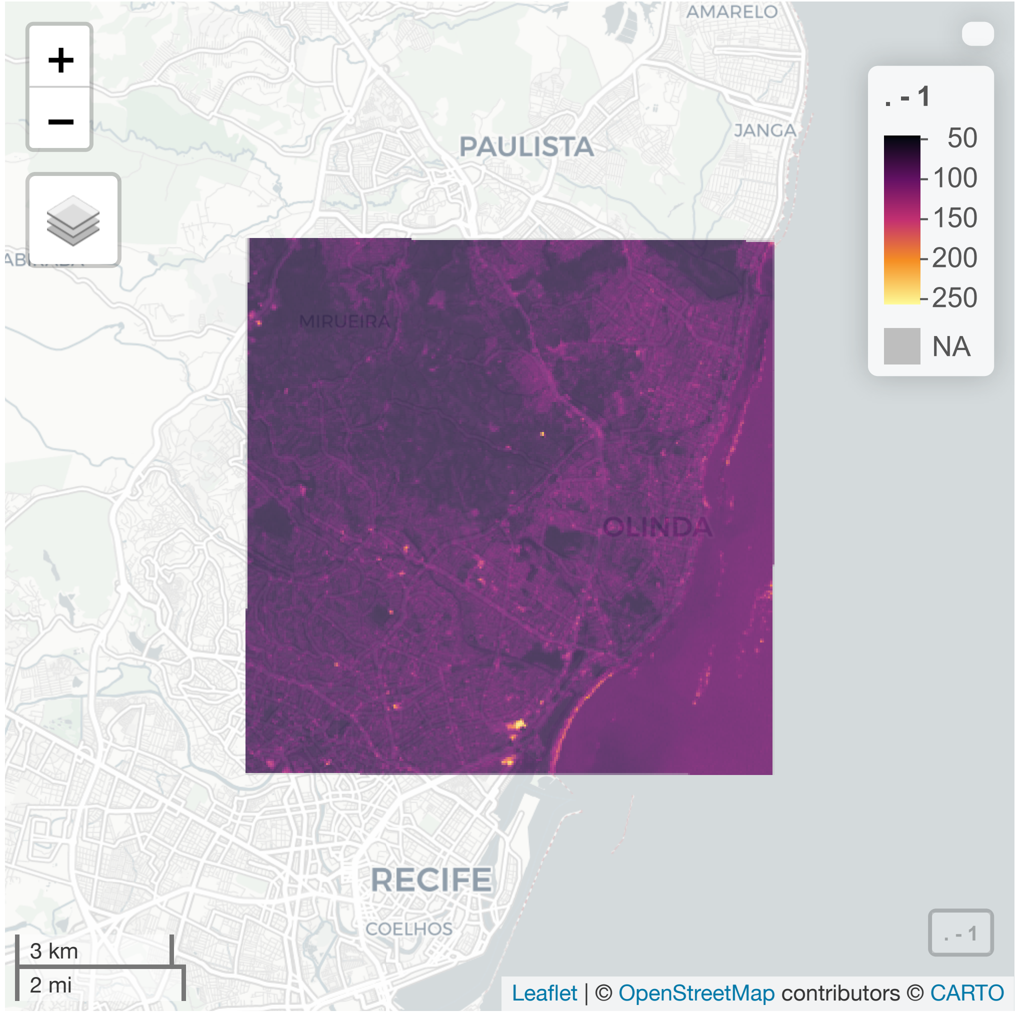
```
## Simple feature collection with 177 features and 2 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -90 xmax: 180 ymax: 83.64513
## CRS:           +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## # A tibble: 177 × 3
##    name                   continent                                     geometry
##    <chr>                  <chr>                               <MULTIPOLYGON [°]>
##  1 Afghanistan            Asia                    (((61.21082 35.65007, 62.2306…
##  2 Angola                 Africa                  (((16.32653 -5.87747, 16.5731…
##  3 Albania                Europe                  (((20.59025 41.8554, 20.46318…
##  4 United Arab Emirates   Asia                    (((51.57952 24.2455, 51.75744
```

What can't {sf} do

The simple features standard does not include raster datasets.

This might include:

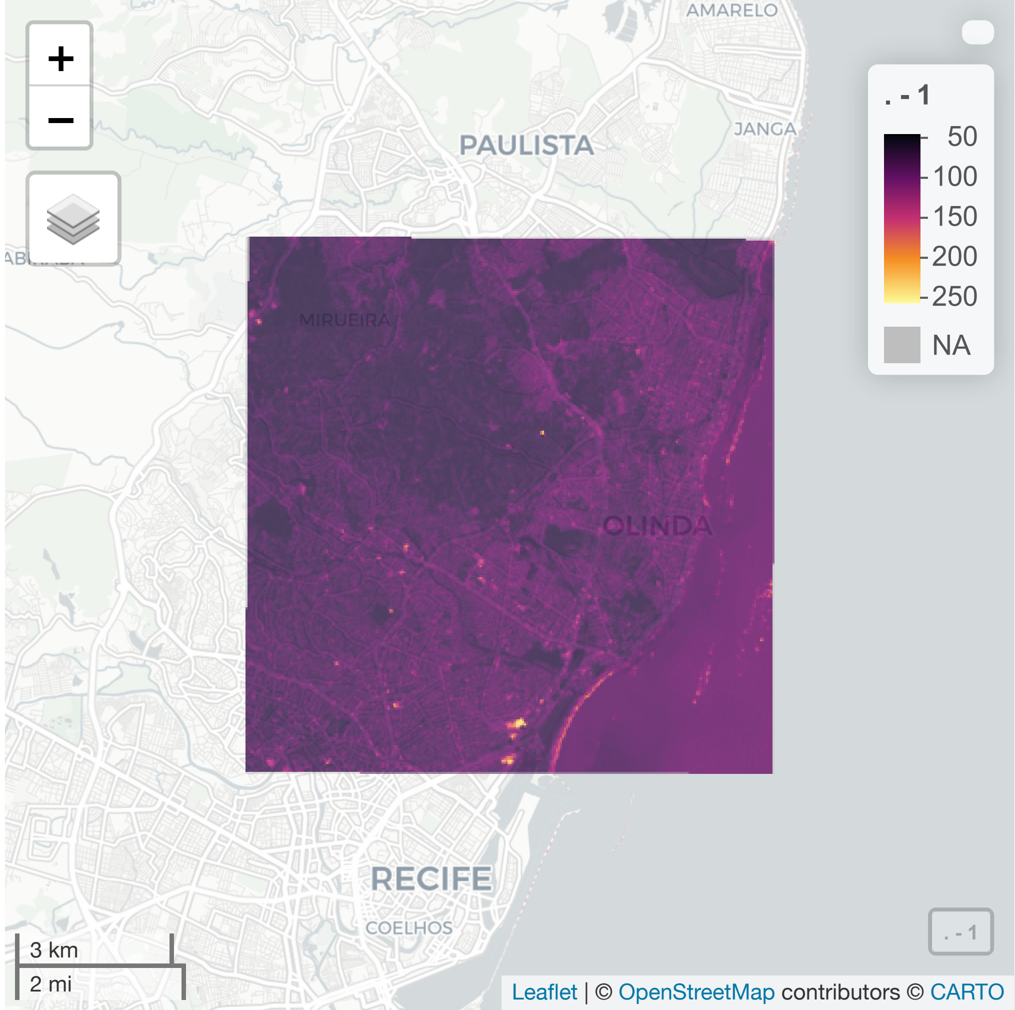
- air pollution data
- meteorological data
- satellite imagery



What can't {sf} do

You'll likely know up front if you're going to work with raster GIS datasets.

We're going to focus on getting vector GIS datasets into R with `{sf}` before introducing `{raster}` and `{stars}` for raster datasets.



(RStudio Coding Slide)

{sf} replaces {sp}

`{sf}` is still a fairly new package, first appearing in 2016.

It is designed to completely replace the older `{sp}` package.

Thankfully, it's very easy to convert `{sp}` objects to `{sf}`.

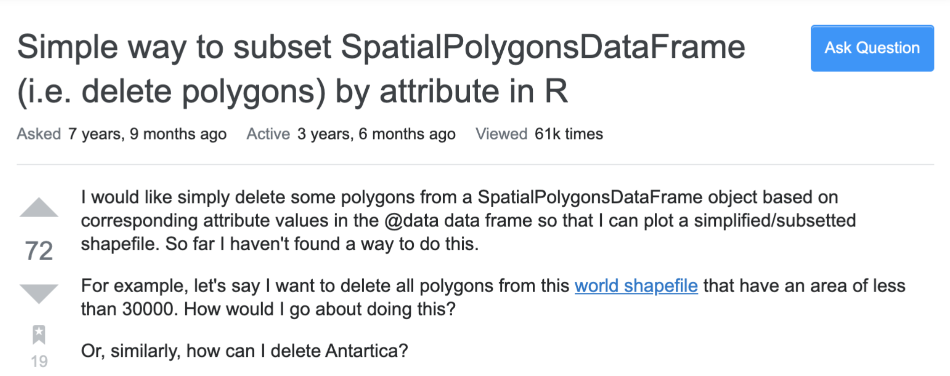
`{sp}` is responsible for the Spatial*DataFrame data sructure, eg:

- SpatialPolygonsDataFrame
- SpatialPointsDataFrame

Working with {sp} blogposts/tutorials

You have a two choices:

- Use the existing `{sp}` code and convert the final output to an `{sf}` object.



Simple way to subset SpatialPolygonsDataFrame (i.e. delete polygons) by attribute in R

Asked 7 years, 9 months ago Active 3 years, 6 months ago Viewed 61k times

I would like simply delete some polygons from a SpatialPolygonsDataFrame object based on corresponding attribute values in the @data data frame so that I can plot a simplified/subsetted shapefile. So far I haven't found a way to do this.

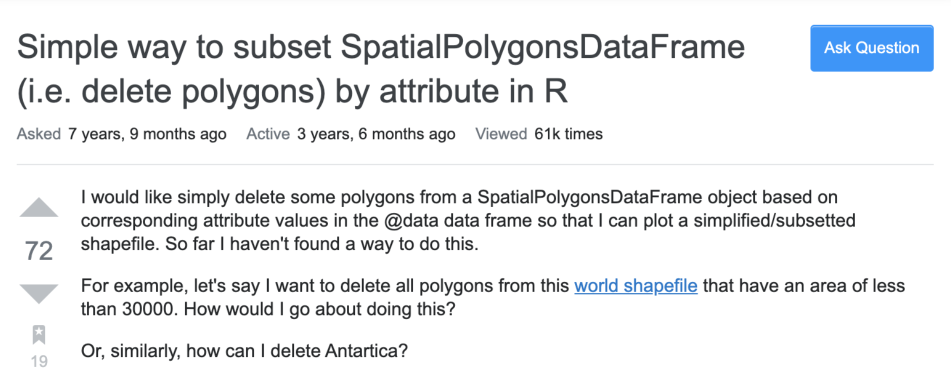
For example, let's say I want to delete all polygons from this world shapefile that have an area of less than 30000. How would I go about doing this?

Or, similarly, how can I delete Antartica?

Working with {sp} blogposts/tutorials

You have a two choices:

- Use the existing `{sp}` code and convert the final output to an `{sf}` object.
- Re-write the code to use `{sf}` throughout. **If you figure it out add a solution or write a blogpost!**



Simple way to subset SpatialPolygonsDataFrame (i.e. delete polygons) by attribute in R

Asked 7 years, 9 months ago Active 3 years, 6 months ago Viewed 61k times

I would like simply delete some polygons from a SpatialPolygonsDataFrame object based on corresponding attribute values in the @data data frame so that I can plot a simplified/subsetted shapefile. So far I haven't found a way to do this.

For example, let's say I want to delete all polygons from this world shapefile that have an area of less than 30000. How would I go about doing this?

Or, similarly, how can I delete Antartica?

(RStudio Coding Slide)

Your turn

Use `mapview()` to visualise only the regions in `tiny_countries110` where the UN region is either "Seven Seas (open ocean)" or "Oceania".

- Load the {tidyverse}, {mapview}, {rnaturalearthdata} and {sf} packages
- Convert `tiny_countries110` into an sf object
- Filter only those countries in "Seven Seas (open ocean)" or "Oceania"
- Visualise this object with {mapview}