Spatial Data in R

Working with simple features

2021-06-17

Outline

1. Setup
2. Simple feature overview
3. Basic plotting
4. Reading and writing
5. Projecting
6. Attribute operations
7. Spatial operations
8. Geometry operations

Setup

- Download and open *simple_features.Rmd*. Full link:
- Install packages

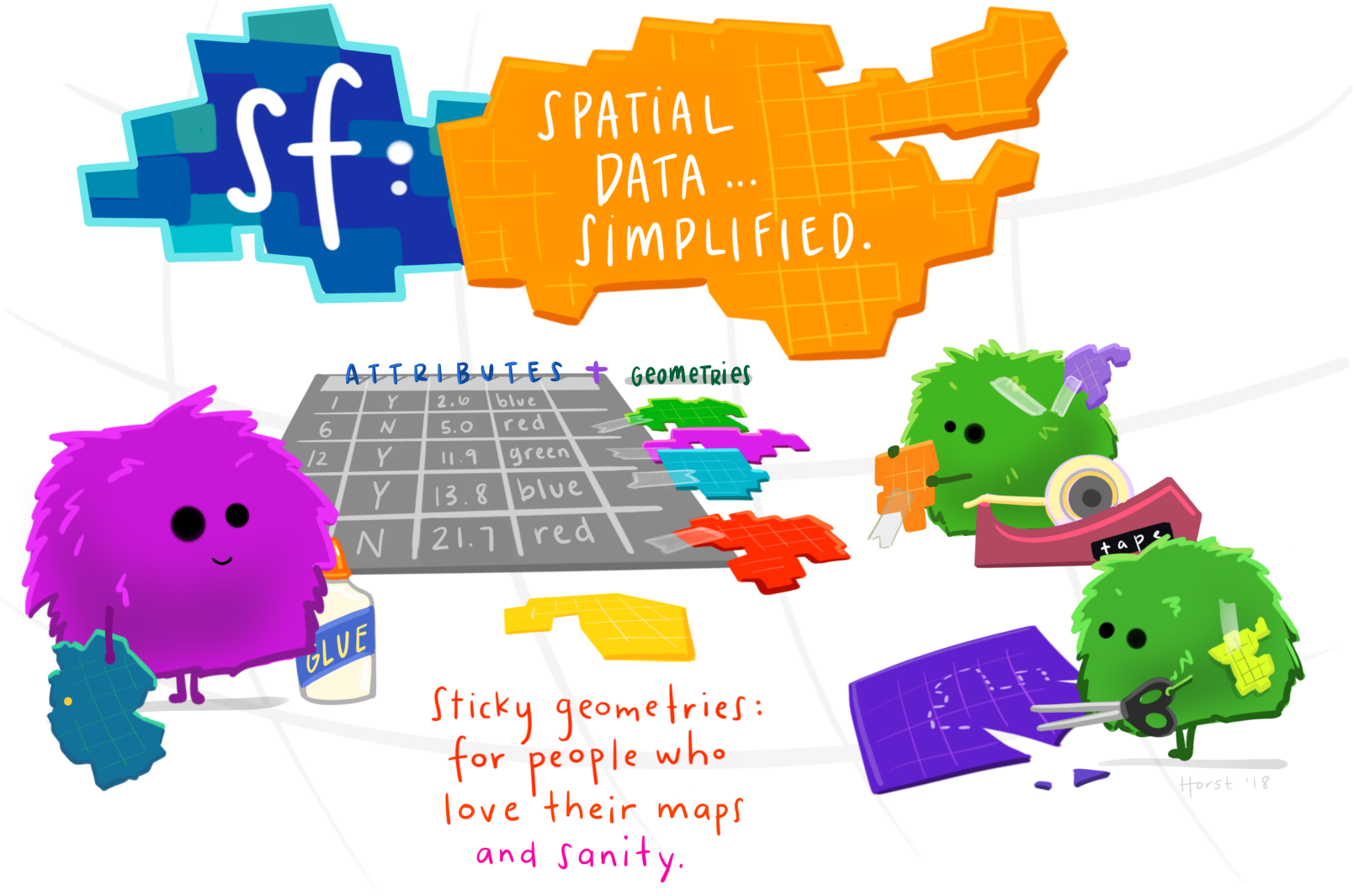
```
install.packages(c("dplyr", "here", "raster", "spData", "sf", "viridis"))

install.packages("spDataLarge",
                 repos = "https://nowosad.github.io/drat/",
                 type = "source")
```

- Attach packages

```
library(raster)
library(spData)
library(spDataLarge)
library(sf)
library(viridis)
library(dplyr)
```

Why `sf`?

1. It's **fast** processing,
2. It's **pretty** maps,
3. It's **rectangular** data,
4. It's **tidy** design, and
5. It's **consistent** syntax, just look for the `st` prefix



Artwork by Allison Horst.

Simple Feature Attributes

```
plot(world["area_km2"], pal = viridis)
```

```
## Error in plot.default(0, type = "n", xlab = "", ylab = "", xaxt = "n", : object 'bb8' not found
```

```
## Error in polypath(p_bind(L), border = border[i], lty = lty[i], lwd = lwd[i], : plot.new has not been called
```

Simple Feature Geometry

```
plot(st_geometry(world), col = "gray90")
```

```
## Error in plot.default(0, type = "n", xlab = "", ylab = "", xaxt = "n", : object 'bb8' not found
```

```
## Error in polypath(p_bind(L), border = border[i], lty = lty[i], lwd = lwd[i], : plot.new has not been called
```

Anatomy of a Simple Feature

```
world
```

```
## Simple feature collection with 177 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -90 xmax: 180 ymax: 83.64513
## Geodetic CRS:  WGS 84
## # A tibble: 177 x 6
##   name_long  continent  area_km2     pop gdpPercap                       geom
##   <chr>      <chr>         <dbl>   <dbl>     <dbl>         <MULTIPOLYGON [°]>
## 1 Fiji       Oceania      1.93e4  8.86e5     8222. (((180 -16.06713, 180 -16.55~
## 2 Tanzania   Africa       9.33e5  5.22e7     2402. (((33.90371 -0.95, 34.07262 ~
## 3 Western S~ Africa       9.63e4 NA            NA  (((-8.66559 27.65643, -8.665~
## 4 Canada     North Ame~   1.00e7  3.55e7    43079. (((-122.84 49, -122.9742 49.~
## 5 United St~ North Ame~   9.51e6  3.19e8    51922. (((-122.84 49, -120 49, -117~
## # ... with 172 more rows
```

simple feature column (sfc)

```
world$geom
```

```
## Geometry set for 177 features
## Geometry type: MULTIPOLYGON
## Dimension:     XY
## Bounding box:  xmin: -180 ymin: -90 xmax: 180 ymax: 83.64513
## Geodetic CRS:  WGS 84
## First 5 geometries:
## MULTIPOLYGON (((180 -16.06713, 180 -16.55522, 1...
## MULTIPOLYGON (((33.90371 -0.95, 34.07262 -1.059...
## MULTIPOLYGON (((-8.66559 27.65643, -8.665124 27...
## MULTIPOLYGON (((-122.84 49, -122.9742 49.00254,...
## MULTIPOLYGON (((-122.84 49, -120 49, -117.0312 ...
```

Every geometry column is a *list*.

simple feature geometry (sfg)

```
world$geom[[1]]
```

```
## MULTIPOLYGON (((180 -16.06713, 180 -16.55522, 1...
```

A single feature's geometry is Well-Known Text.

From the ground up

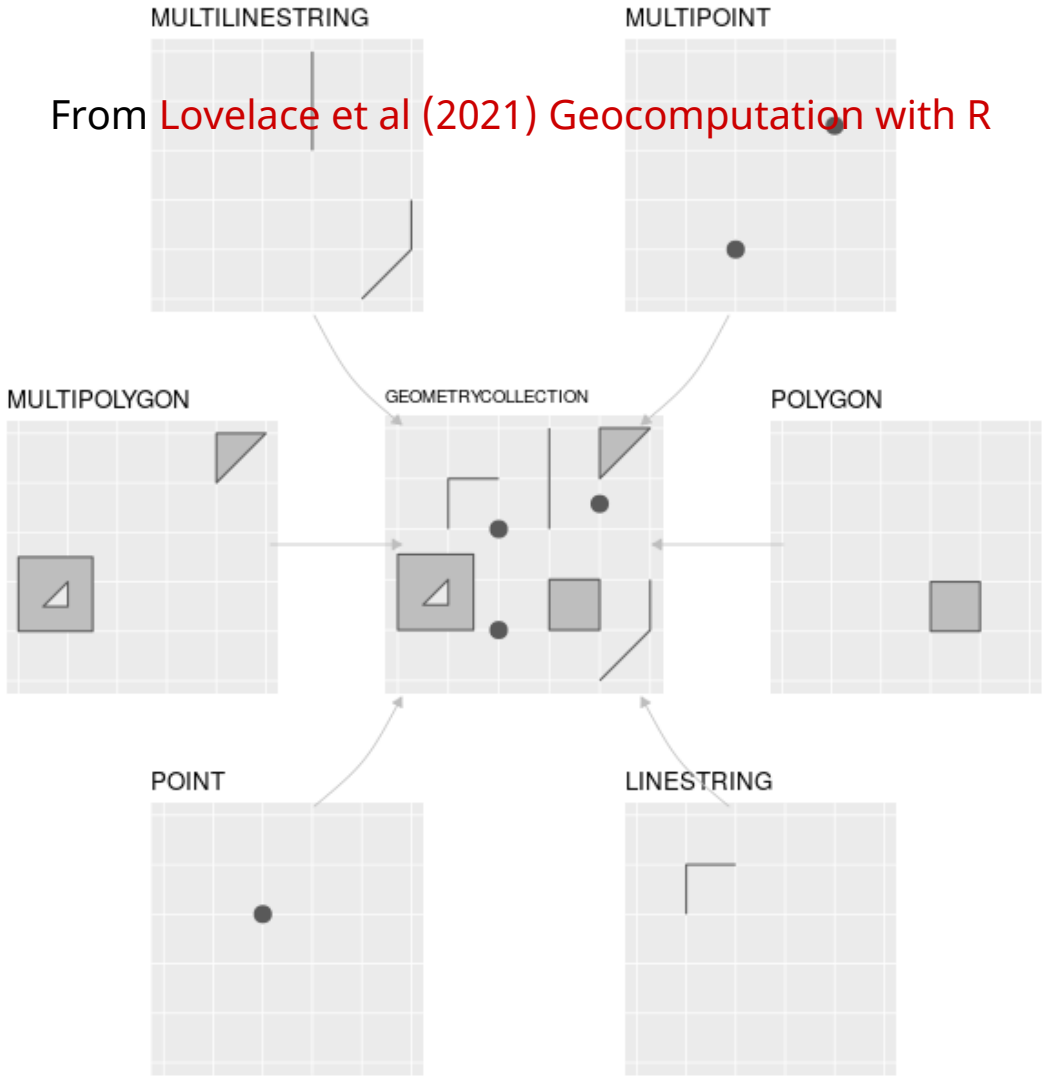
```
point1 <- st_point(c(432000, 4513100))
point2 <- st_point(c(436750, 4518500))

sf_column <- st_sfc(point1, point2, crs = 26912)

st_sf(id = 1:2,
      location = "red butte canyon",
      geometry = sf_column)
```

```
## Simple feature collection with 2 features and 2 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 432000 ymin: 4513100 xmax: 436750 ymax: 4518500
## Projected CRS: NAD83 / UTM zone 12N
##   id         location               geometry
## 1  1 red butte canyon POINT (432000 4513100)
## 2  2 red butte canyon POINT (436750 4518500)
```

Similar to `data.frame(col1 = ..., col2 = ...)`, but with a geometry column.

Geometry types

From Lovelace et al (2021) Geocomputation with R

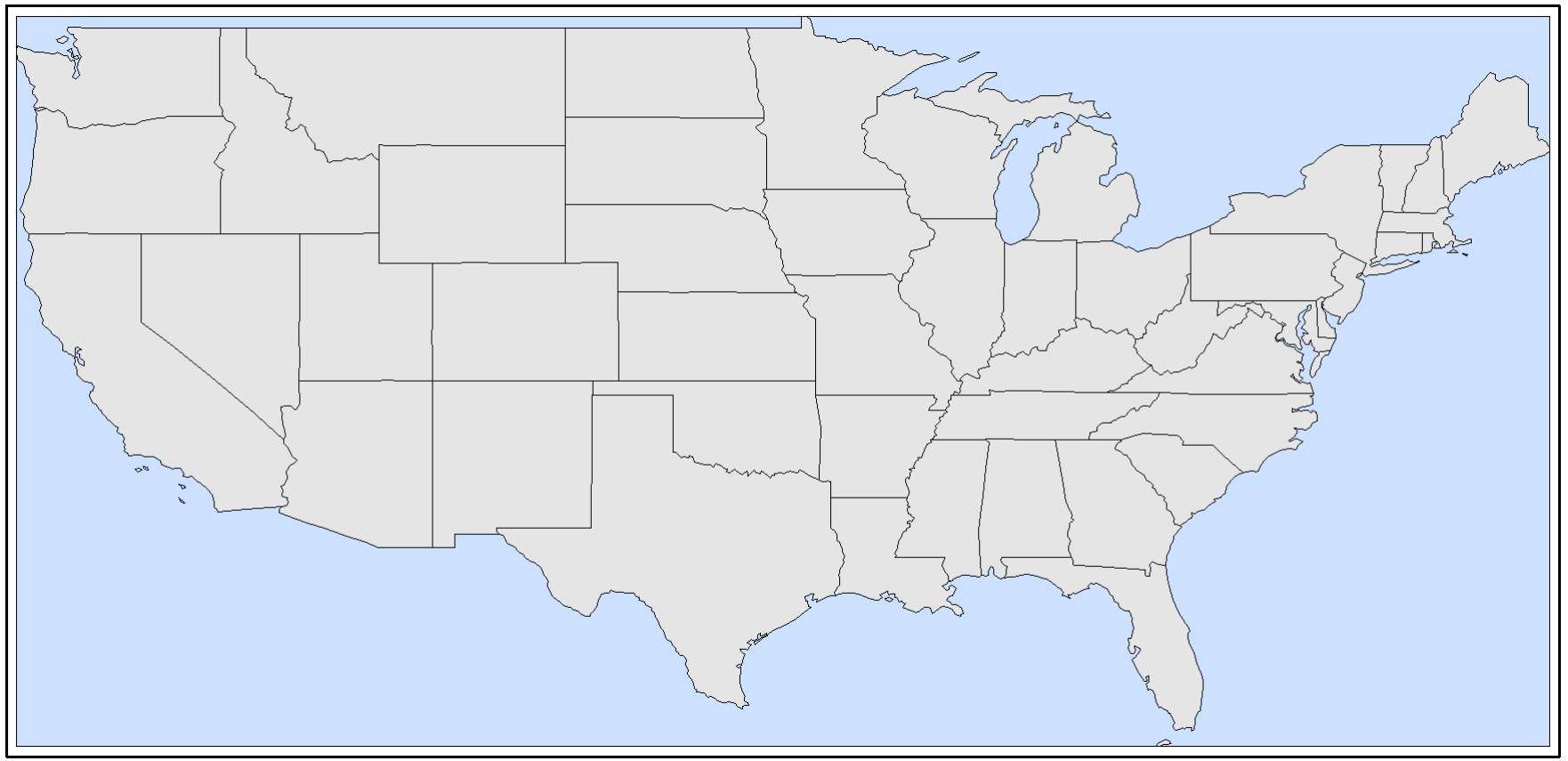


Projecting simple features

```
st_transform(world, crs = 3857)
```

The easiest way to specify a CRS is to use the EPSG code. In this example, that's *3857*, which refers to the Google Mercator reference system.

Writing vectors



Writing vectors

```
st_write(us_states, dsn = here("gis", "us_states.shp"))
```

dsn is short for **d**ata **s**ource **n**ame, can be a file, a folder, a local database, or a connection to a remote database.