Spatial data in R: simple features and future perspectives

Edzer Pebesma (ifgi, Münster, DE)
Roger Bivand (NHH, Bergen, NO)

UseR! Stanford, Jun 27-30, 2016

What are simple features?

First: what is meant by a *feature*?

- any *thing* in the (real) world
- persons, cars, buildings, rivers, mountains, ...
- but also surfaces, and collections of all of these

Simple features refer to:

- a *common architecture for simple feature geometry*
- *a formal standard*: OGC 06-103r4; ISO 19125:
- "OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture"
- a set of encodings:
  - WKT: "well known text"
  - WKB: "well known binary"

What are simple features?

First: what is meant by a *feature*?

- any *thing* in the (real) world
- persons, cars, buildings, rivers, mountains, ...
- but also surfaces, and collections of all of these

Simple features refer to:

- a *common architecture for simple feature geometry*
- *a formal standard*: OGC 06-103r4; ISO 19125:
- "OpenGIS Implementation Standard for Geographic information - Simple feature access - Part 1: Common architecture"
- a set of encodings:
  - WKT: "well known text"
  - WKB: "well known binary"

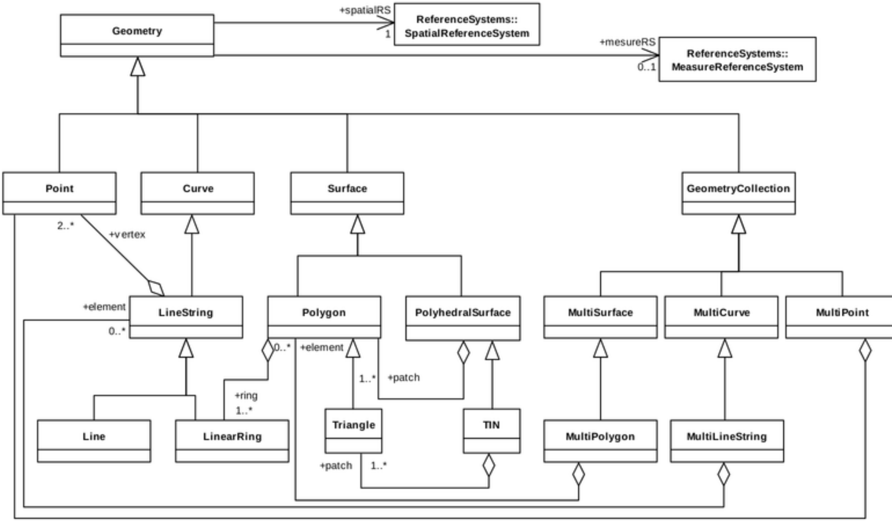


Figure 1: Geometry class hierarchy

How do simple features look like?

Encoded as well-known-text:

```
POINT(0 0)
LINESTRING(0 0,1 1,1 2)
POLYGON((0 0,4 0,4 4,0 4,0 0),(1 1, 2 1, 2 2, 1 2,1 1))
MULTIPOINT((0 0),(1 2))
MULTILINESTRING((0 0,1 1,1 2),(2 3,3 2,5 4))
MULTIPOLYGON(((0 0,4 0,4 4,0 4,0 0),(1 1,2 1,2 2,1 2,1 1)),
    ((-1 -1,-1 -2,-2 -2,-2 -1,-1 -1)))
GEOMETRYCOLLECTION(POINT(2 3),LINESTRING(2 3,3 4))
```

Polygons:

- first polygon: enclosing, counter-clockwise
- second, third, ... polygons: holes, clockwise

2D-only?

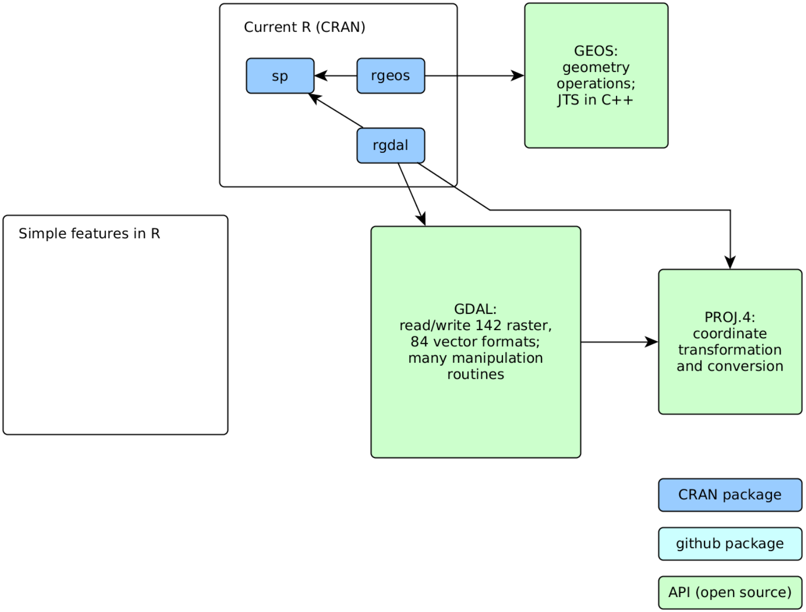
No:

```
POINT Z(0 0 0)
POINT M(0 0 0)
POINT ZM(0 0 0 0)
LINESTRING Z(0 0 1,1 1 1,1 2 3)
POLYGON M((0 0 1,4 0 0,4 4 2,0 4 1,0 0 1))
```

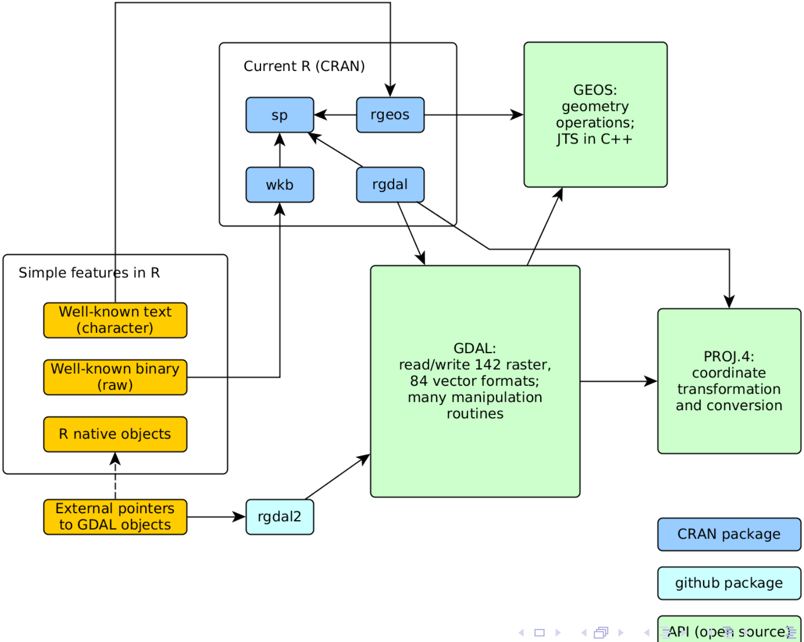
- `Z`: third spatial dimension (altitude, height)
- `M`: "measure": "A Point value may include an m coordinate value. The m coordinate value allows the application environment to associate some measure with the point values. For example: A stream network may be modeled as multilinestring value with the m coordinate values measuring the distance from the mouth of stream. "

`M` cannot be thought of as usual attributes of a polygon or line: an `M` value is associated with each *point* of a polygon, line, ...

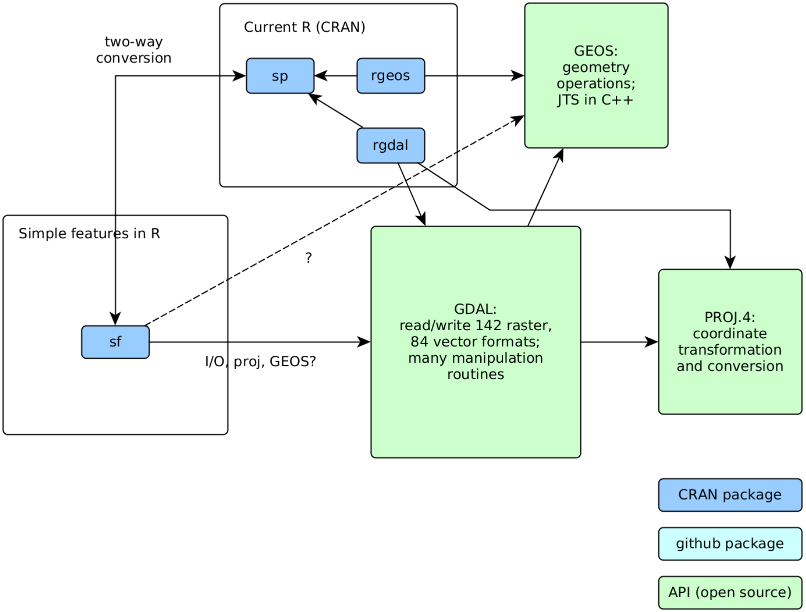
Current situation in R, w/o SF



Current, with SF pieces



Where we want to go



Simple features in R: a proposal

We usually work with sets of simple features, where feature properties (attributes) are in a `data.frame` or similar.
Useful constraints will be:

- sets will have a single type (which can, in case of a mix, be `GEOMETRYCOLLECTION`)
- sets will have a single coordinate reference system

Keep it simple:

- feature sets should be a list, and work as a list column in `data.frame` and the like (tidy!)
- use `numeric` for single point, `matrix` for a set of points, `list` for set of sets
- use S3
- of class `sf`, attributes `type` (chr), `epsg` (int) and `proj4string` (chr)

"list column"

```
> (d = data.frame(a = 1:3, b = I(list(1:2, c(1,3,5), 10:5))))

  a            b
1 1         1, 2
2 2      1, 3, 5
3 3 10, 9, 8....

> summary(d)

       a        b.Length  b.Class  b.Mode
 Min.   :1.0   2         -none-   numeric
 1st Qu.:1.5   3         -none-   numeric
 Median :2.0   6         -none-   numeric
 Mean   :2.0
 3rd Qu.:2.5
 Max.   :3.0

> library(tibble)
> data_frame(a = 1:3, b = list(1:2, c(1,3,5), 10:5))

Source: local data frame [3 x 2]

      a         b
  <int>    <list>
1     1 <int [2]>
2     2 <dbl [3]>
3     3 <int [6]>
```

R implementation: proposal

Although 7 of them are dominant, there are 72 types:

XY	XYZ	XYM	XYZM
Geometry	Geometry Z	Geometry M	Geometry ZM
Point	Point Z	Point M	Point ZM
LineString	LineString Z	LineString M	LineString ZM
Polygon	Polygon Z	Polygon M	Polygon ZM
MultiPoint	MultiPoint Z	MultiPoint M	MultiPoint ZM
MultiLineString	MultiLineString Z	MultiLineString M	MultiLineString ZM
MultiPolygon	MultiPolygon Z	MultiPolygon M	MultiPolygon ZM
GeometryCollection	GeometryCollection Z	GeometryCollection M	GeometryCollection ZM
CircularString	CircularString Z	CircularString M	CircularString ZM
CompoundCurve	CompoundCurve Z	CompoundCurve M	CompoundCurve ZM
CurvePolygon	CurvePolygon Z	CurvePolygon M	CurvePolygon ZM
MultiCurve	MultiCurve Z	MultiCurve M	MultiCurve ZM
MultiSurface	MultiSurface Z	MultiSurface M	MultiSurface ZM
Curve	Curve Z	Curve M	Curve ZM
Surface	Surface Z	Surface M	Surface ZM
PolyhedralSurface	PolyhedralSurface Z	PolyhedralSurface M	PolyhedralSurface ZM
TIN	TIN Z	TIN M	TIN ZM
Triangle	Triangle Z	Triangle M	Triangle ZM

How does a spatial table look, in PostGIS?

```
edzer@gin-edzer:~$ psql postgis
psql (9.3.13)
Type "help" for help.

postgis=# select * from meuse2 limit 2;
 id | zinc |                          geom
----+------+----------------------------------------------------
  1 | 1022 | 0101000020E6100000000000000804640000000000000804640
  2 | 1141 | 01010000204071000000000000819064100000000D85B1441
(2 rows)

postgis=# select zinc, ST_asText(geom) from meuse2 limit 2;
 zinc |      st_astext
------+----------------------
 1022 | POINT(181072 333611)
 1141 | POINT(181025 333558)
(2 rows)
```

PostGIS keeps in two other tables the information

- that `meuse2` has geometry column `geom`, the CRS ID of it
- what this CRS ID refers to (proj4string, WKT of CRS)

Reading WKT through DBI/RPostgreSQL

```
> library(RPostgreSQL)
> drv <- dbDriver("PostgreSQL")
> con <- dbConnect(drv,
+  dbname="postgis", user="edzer", password="pw",
+  host="localhost", port='5432')
> query = "select zinc, geom from meuse2 limit 2;"
> (tbl = fetch(dbSendQuery(con, query)))

  zinc                                               geom
1 1022 0101000020407100000000000000801A064100000000AC5C1441
2 1141 010100002040710000000000000081906410000000D85B1441


Warning message:
In postgresqlExecStatement(conn, statement, ...) :
  RS-DBI driver warning: (unrecognized PostgreSQL field type geometry (id:16393) in column 1)


> sapply(tbl, class)

       zinc        geom
  "numeric" "character"

> query = "select zinc, ST_asText(geom) from meuse2 limit 2;"
> (tbl = fetch(dbSendQuery(con, query)))

  zinc               st_astext
1 1022 POINT(181072 333611)
2 1141 POINT(181025 333558)

> sapply(tbl, class)

       zinc   st_astext
  "numeric" "character"
```

`sf`: design considerations (1/2)

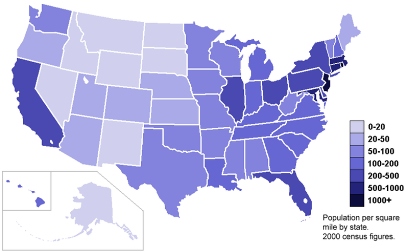
- read + write using external libraries (GDAL)
- support PROJ.4 compatible CRS handling
- CRS transformation/conversion through GDAL (= PROJ.4)
- "stick" to S3
- single SF items shall have a class: `sfi`, or `POINT`, `POLYGON` etc
- sets of SF (list column) shall have a class `sfc`, and have `bbox` and `CRS` attributes
- `sf` table objects with a *single* `sfc` shall have a class: `sf`
- `sf` shall extend its base class:

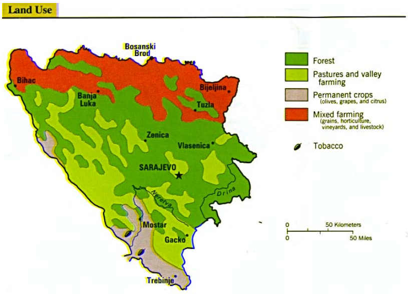
```
> a = data.frame(x = 1:3)
> (class(a) = c("sf", class(a)))

[1] "sf"         "data.frame"
```

- balance simplicity with `sp` compatibility
- use `numeric` for single point, `matrix` for a set of points, `list` for set of sets

`sf`: design considerations (1/2)

- start with the low-hanging fruit of the 2D (XY) geometries `POINT`, `MULTIPOINT`, `LINESTRING`, `POLYGON`, `MULTILINESTRING`, `MULTIPOLYGON`, `GEOMETRYCOLLECTION`
- keep the path open for all 68 SF types (inherit: `XY` $\Rightarrow$ `XYZ`, `XYM` $\Rightarrow$ `XYZM`)
- add functions that convert `sfi` into the arguments needed by `grid::polygonGrob` and the like.
- document for each of the non-spatial variables how it relates to the spatial features (constant, aggregate, NA)


0-20
20-50
50-100
100-200
200-500
500-1000
1000+
Population per square mile by state.
2000 census figures.


Land Use
Bosanski Brod
Bihac
Banja Luka
Bijeljina
Tuzla
Zenica
Vlasenica
SARAJEVO
Neretva
Drina
Mostar
Gacko
Trebinje
Forest
Pastures and valley farming
Permanent crops (olives, grapes, and citrus)
Mixed farming (grains, horticulture, vineyards, and livestock)
Tobacco
0 50 Kilometers
0 50 Miles

Discussion

- it is time for simple features in R; package `sf` will be doing this
- simple features are standard and ubiquitous (databases, geojson, leaflet, ...)
- we found support by R consortium; positive feedback from ESRI too
- now that *list columns* are tidy, so are we
- `sf` will focus on I/O, interoperability, and functionality
  - with R plot methods (base, grid)
  - external data sources (GDAL)
  - geometry operations (intersections etc.)
  - migration path, conversion to/from `sp`