Simple Features in R

Jonathan de Bruin

5/19/2019

Simple Features

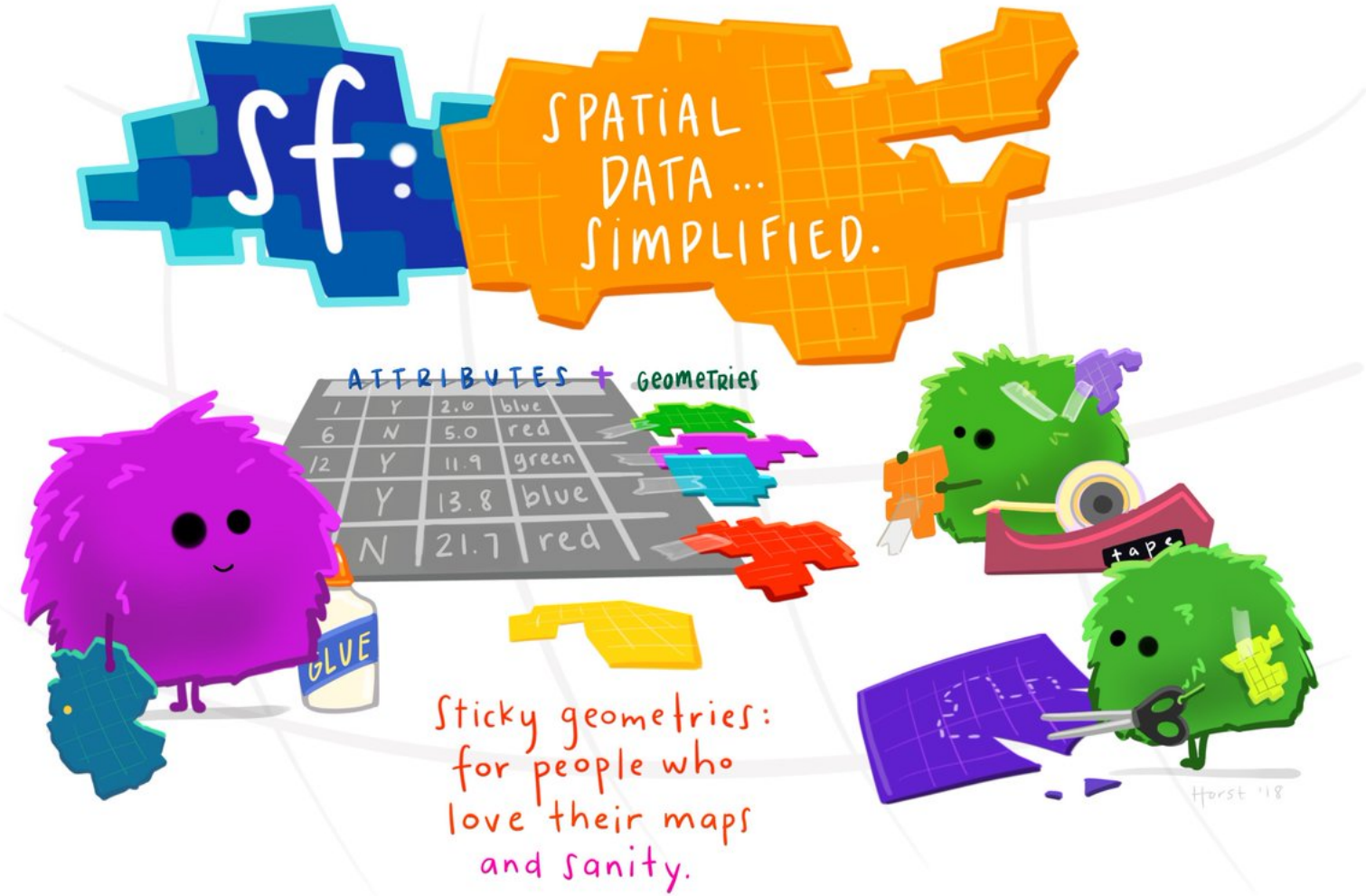


Figure 1: Illustration (c) by Allison Horst

Simple Features

- Simple Feature Access
- ISO 19125-1 and ISO 19125-2
- 17 simple feature types (like point, line, polygon)

> *"A simple feature is defined by the OpenGIS Abstract specification to have both spatial and non-spatial attributes. Spatial attributes are geometry valued, and simple features are based on 2D geometry with linear interpolation between vertices."*

More and more databases and software support Simple Features

- ArcGIS, QGIS
- SQL, Elasticsearch, MongoDB, ...
- R with the package `sf`

Simple Features in R

- Implemented in the package `sf`.
- Provides simple features in data.frames or tibbles with a geometry list-column
- all 17 simple feature types for all dimensions (XY, XYZ, XYM, XYZM)
- https://github.com/r-spatial/sf
- Good replacement for `sp` and `gdal` packages.

```
library(sf)
```

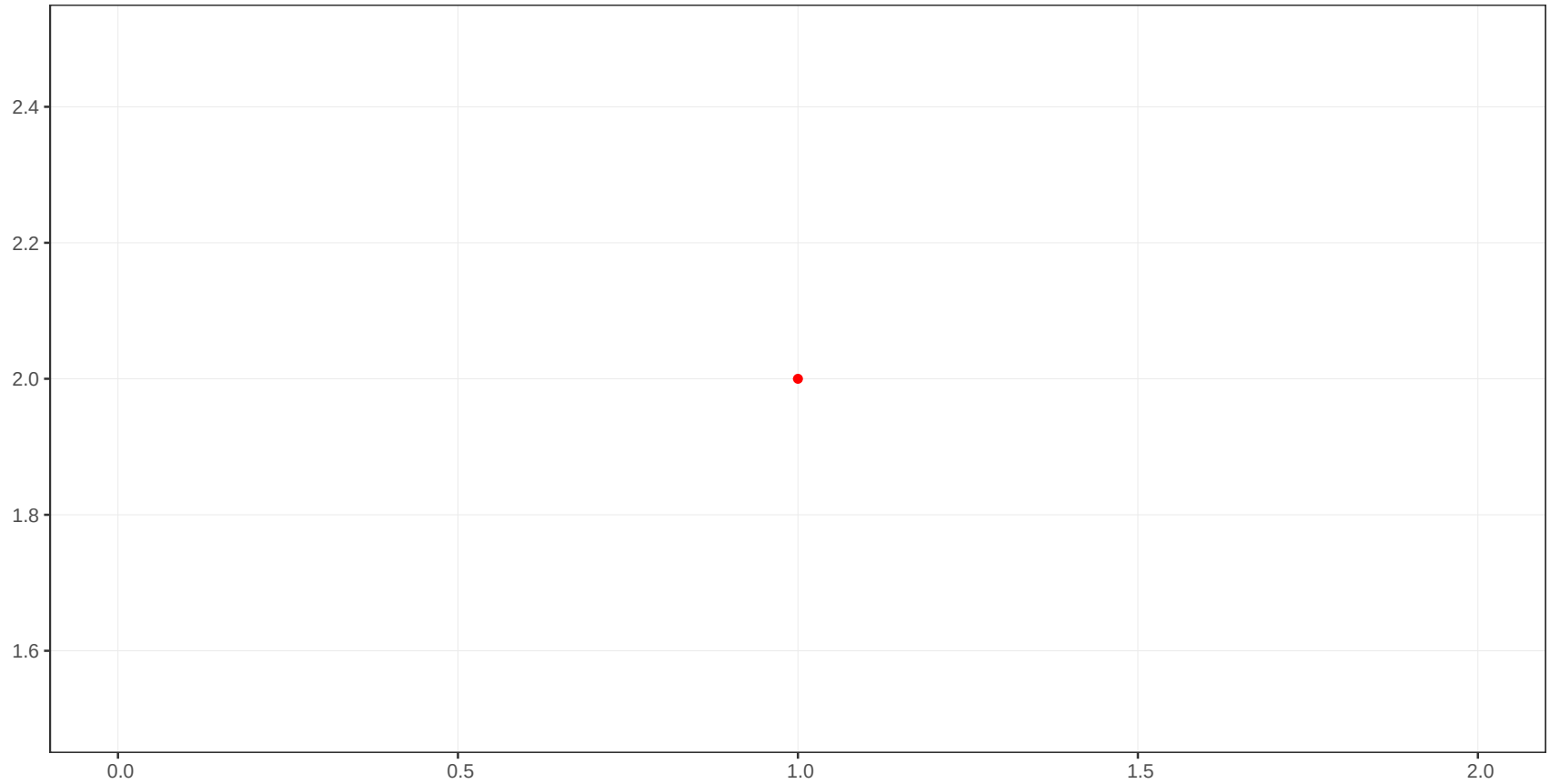
```
## Linking to GEOS 3.6.1, GDAL 2.1.3, PROJ 4.9.3
```

> *"All functions and methods in sf that operate on spatial data are prefixed by `st_`, which refers to spatial and temporal; this makes them easily findable by command-line completion."*

Simple Feature: POINT

```
st_point(c(1,2))
```

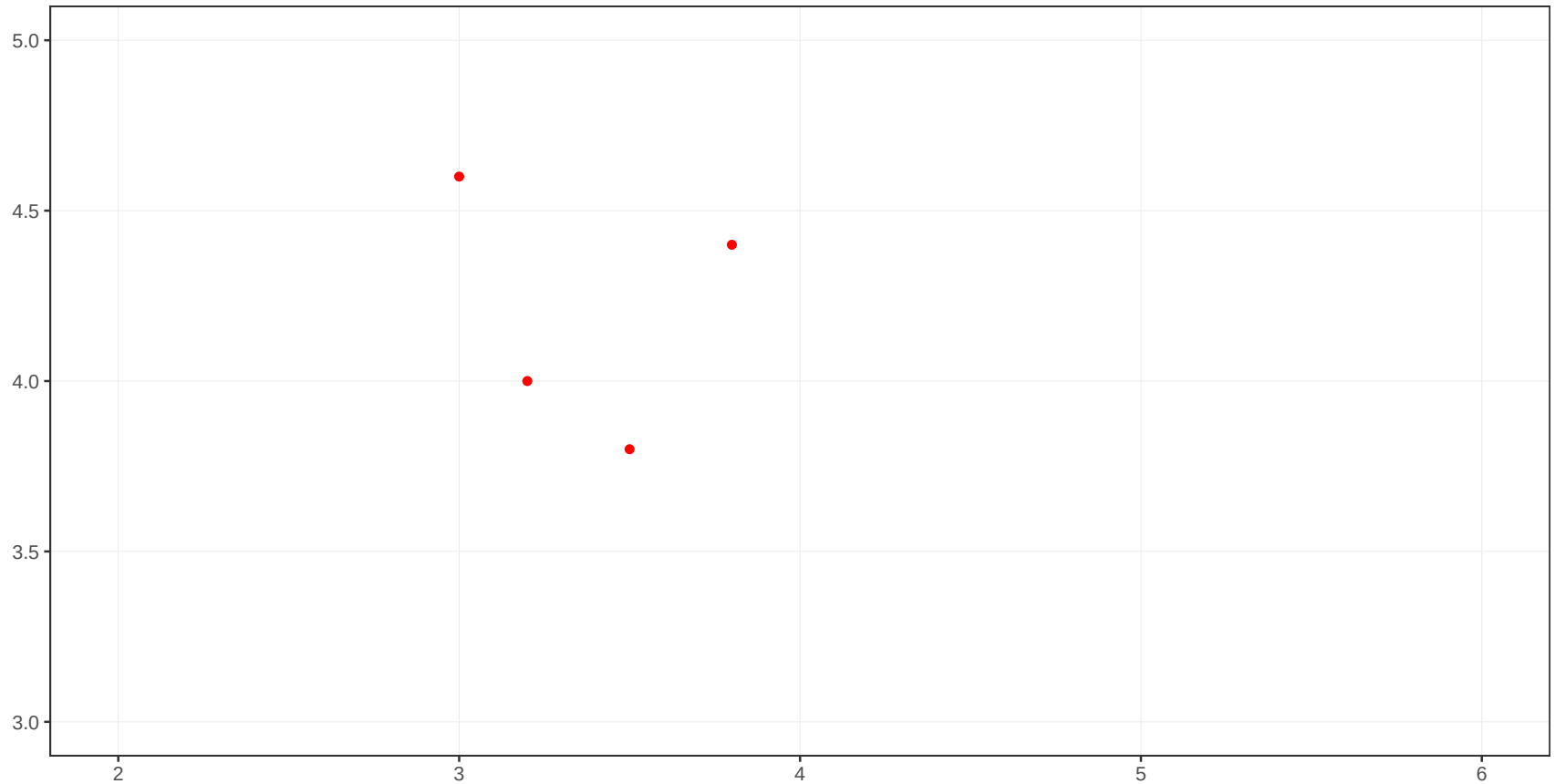
```
## POINT (1 2)
```



Simple Feature: MULTIPOINT

```
p <- rbind(c(3.2,4), c(3,4.6), c(3.8,4.4), c(3.5,3.8))
st_multipoint(p)
```

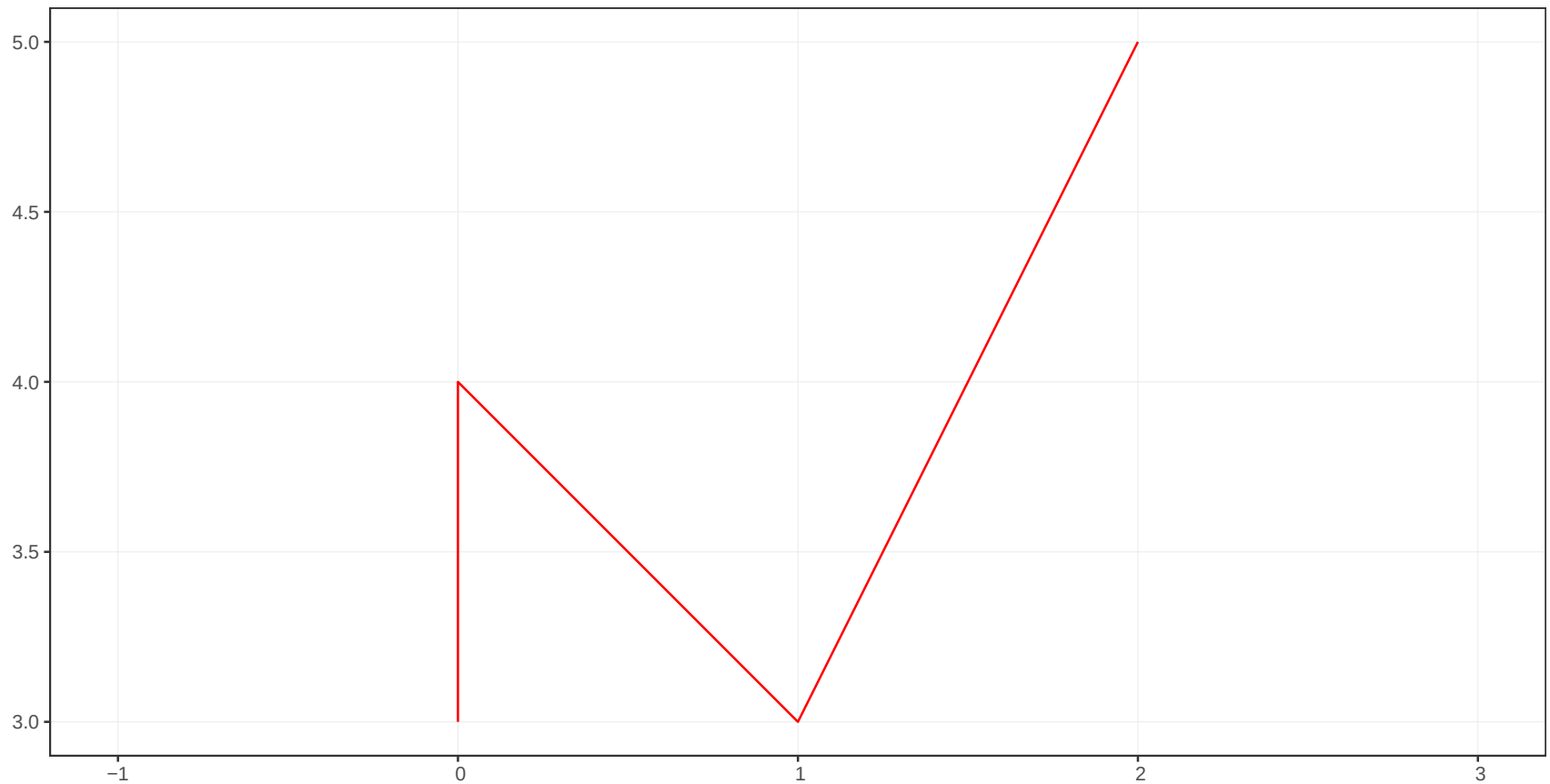
```
## MULTIPOINT (3.2 4, 3 4.6, 3.8 4.4, 3.5 3.8)
```



Simple Feature: LINESTRING

```
s1 <- rbind(c(0,3),c(0,4),c(1,3),c(2,5))
st_linestring(s1)
```

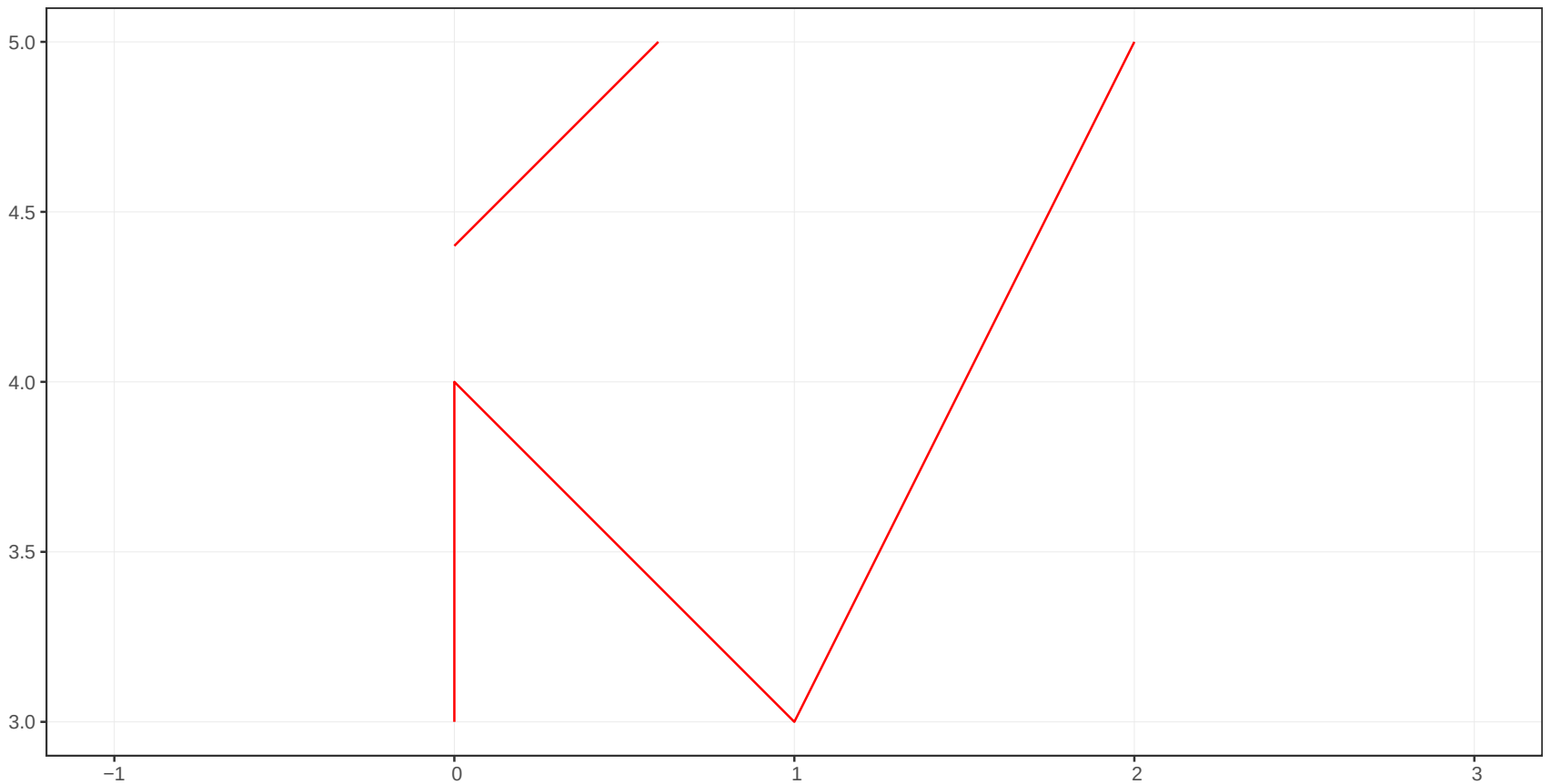
```
## LINESTRING (0 3, 0 4, 1 3, 2 5)
```



Simple Feature: MULTILINESTRING

```
s1 <- rbind(c(0,3),c(0,4),c(1,3),c(2,5))
s2 <- rbind(c(0,4.4), c(0.6,5))
st_multilinestring(list(s1,s2))
```

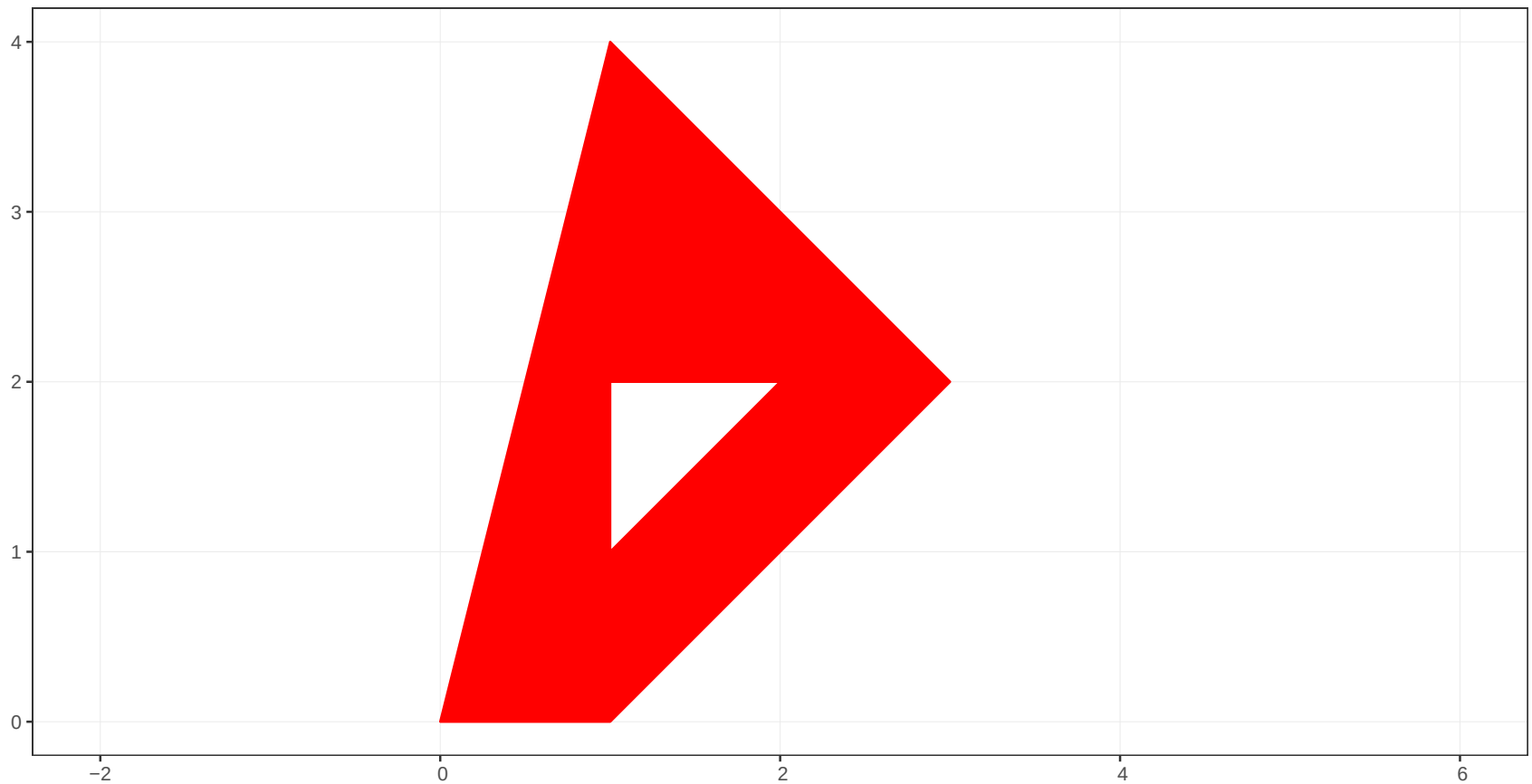
```
## MULTILINESTRING ((0 3, 0 4, 1 3, 2 5), (0 4.4, 0.6 5))
```



Simple Feature: POLYGON

```
p <- rbind(c(0,0), c(1,0), c(3,2), c(1,4), c(0,0))
h <- rbind(c(1,1), c(1,2), c(2,2), c(1,1))
st_polygon(list(p,h))
```

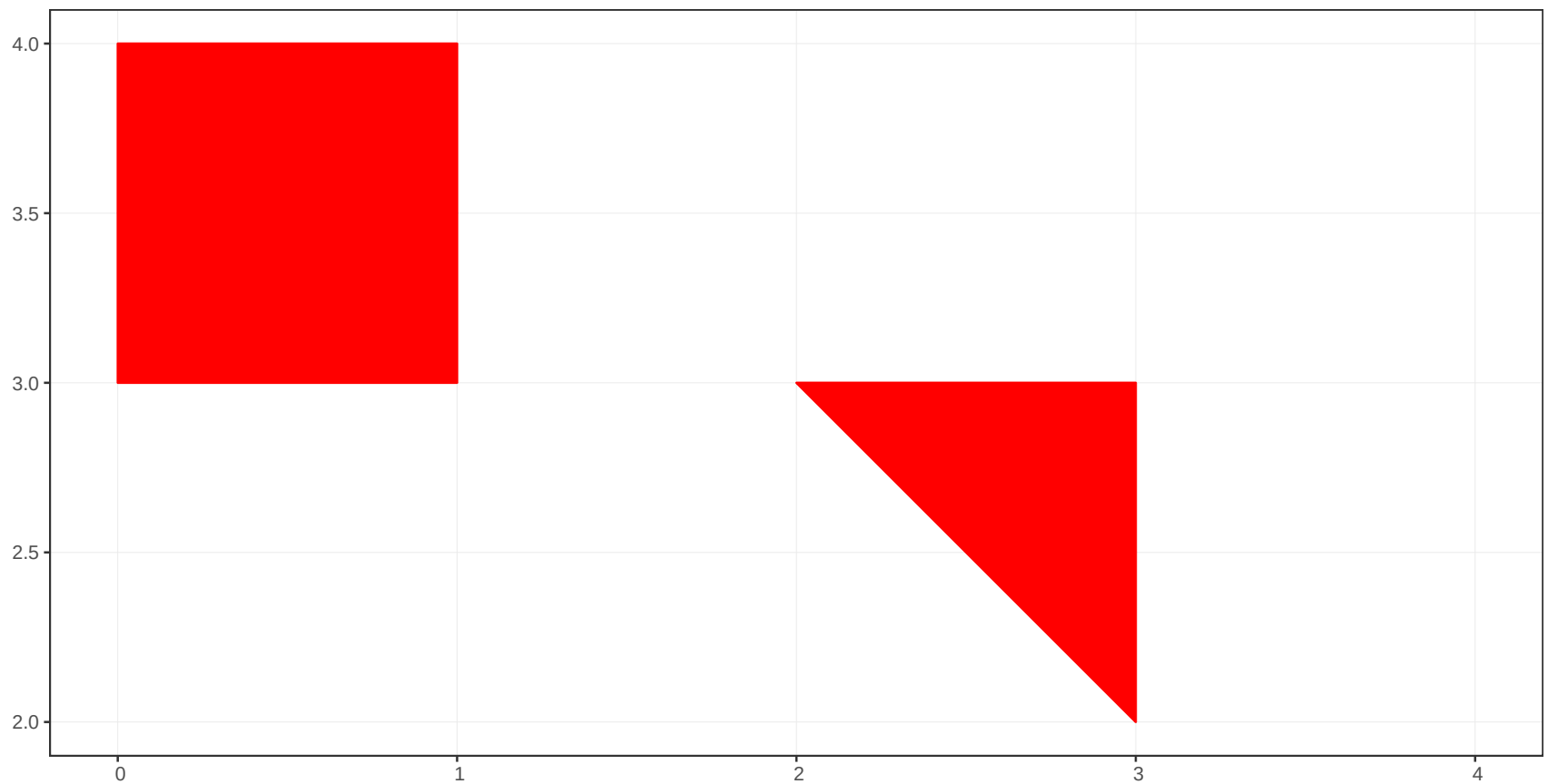
```
## POLYGON ((0 0, 1 0, 3 2, 1 4, 0 0), (1 1, 1 2, 2 2, 1 1)
```



Simple Feature: MULTIPOLYGON

```
p1 <- rbind(c(0,3), c(0,4), c(1,4), c(1,3), c(0,3))
p2 <- rbind(c(3,2), c(2,3), c(3,3), c(3,2))
st_multipolygon(list(list(p1), list(p2)))
```

```
## MULTIPOLYGON (((0 3, 0 4, 1 4, 1 3, 0 3)), ((3 2, 2 3, 3
```



Spatial operations

Geometric Confirmation:

- `st_overlaps`, `st_contains`, `st_disjoint`

Geometric Operations

- `st_centroid`, `st_convex_hull`, `st_line_merge`

Geometry Operations

- `st_intersection`, `st_difference`, `st_union`

Geometric measurement

- `st_distance`, `st_area`

Spatial operations - Intersection

Define two polygons a and b.

```
(a <- st_polygon(list(rbind(
  c(0, 0), c(0, -1), c(7.5, -1), c(7.5, 0), c(0, 0)
))))
```

```
## POLYGON ((0 0, 0 -1, 7.5 -1, 7.5 0, 0 0))
```

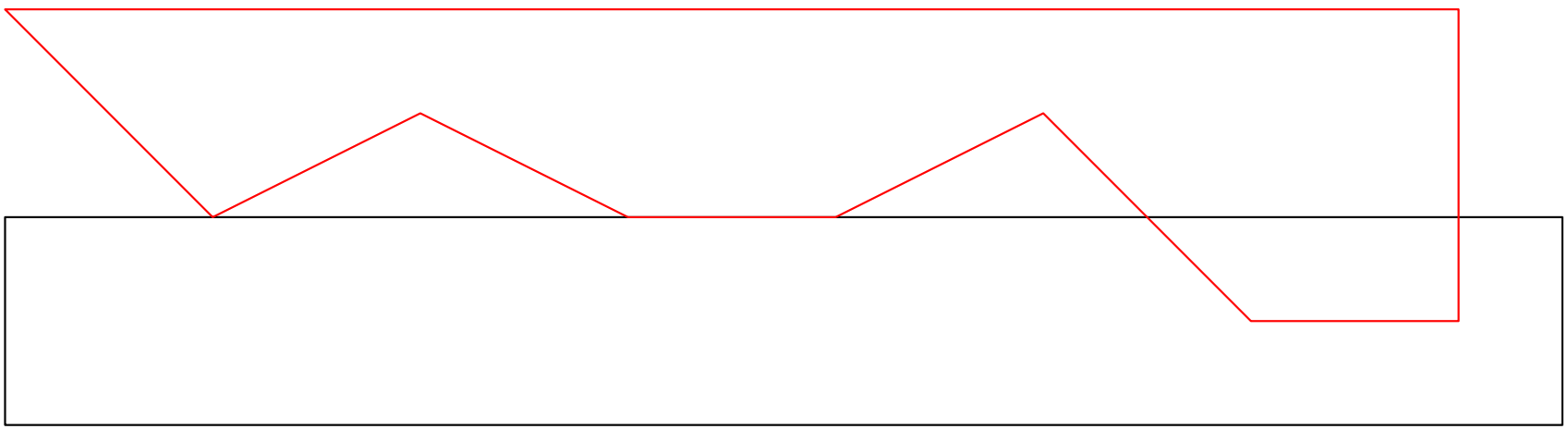
```
(b <- st_polygon(list(rbind(
  c(0,1), c(1,0), c(2,.5), c(3,0), c(4,0),
  c(5,0.5), c(6,-0.5), c(7,-0.5), c(7,1), c(0,1)
))))
```

```
## POLYGON ((0 1, 1 0, 2 0.5, 3 0, 4 0, 5 0.5, 6 -0.5, 7 -0
```

Spatial operations - Intersection

Define two polygons `a` and `b`.

```
plot(a, ylim = c(-1,1))
plot(b, add = TRUE, border = 'red')
```



Spatial operations - Intersection

```
int_a_and_b <- st_intersection(a,b)
int_a_and_b
```

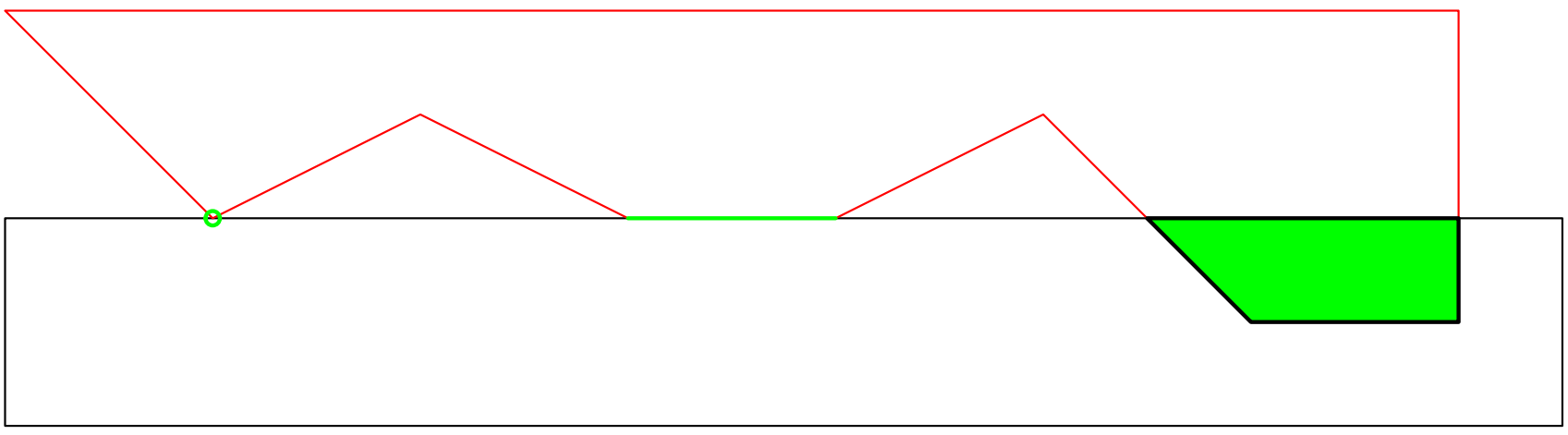
```
## GEOMETRYCOLLECTION (POINT (1 0), LINESTRING (4 0, 3 0),
```

GEOMETRYCOLLECTION

- POINT (1 0)
- LINESTRING (4 0, 3 0)
- POLYGON ((5.5 0, 7 0, 7 -0.5, 6 -0.5, 5.5 0))

Spatial operations - Intersection

```
plot(a, ylim = c(-1,1))
plot(b, add = TRUE, border = 'red')
plot(int_a_and_b, add = TRUE, col = 'green', lwd = 2)
```



Simple features and data.frames

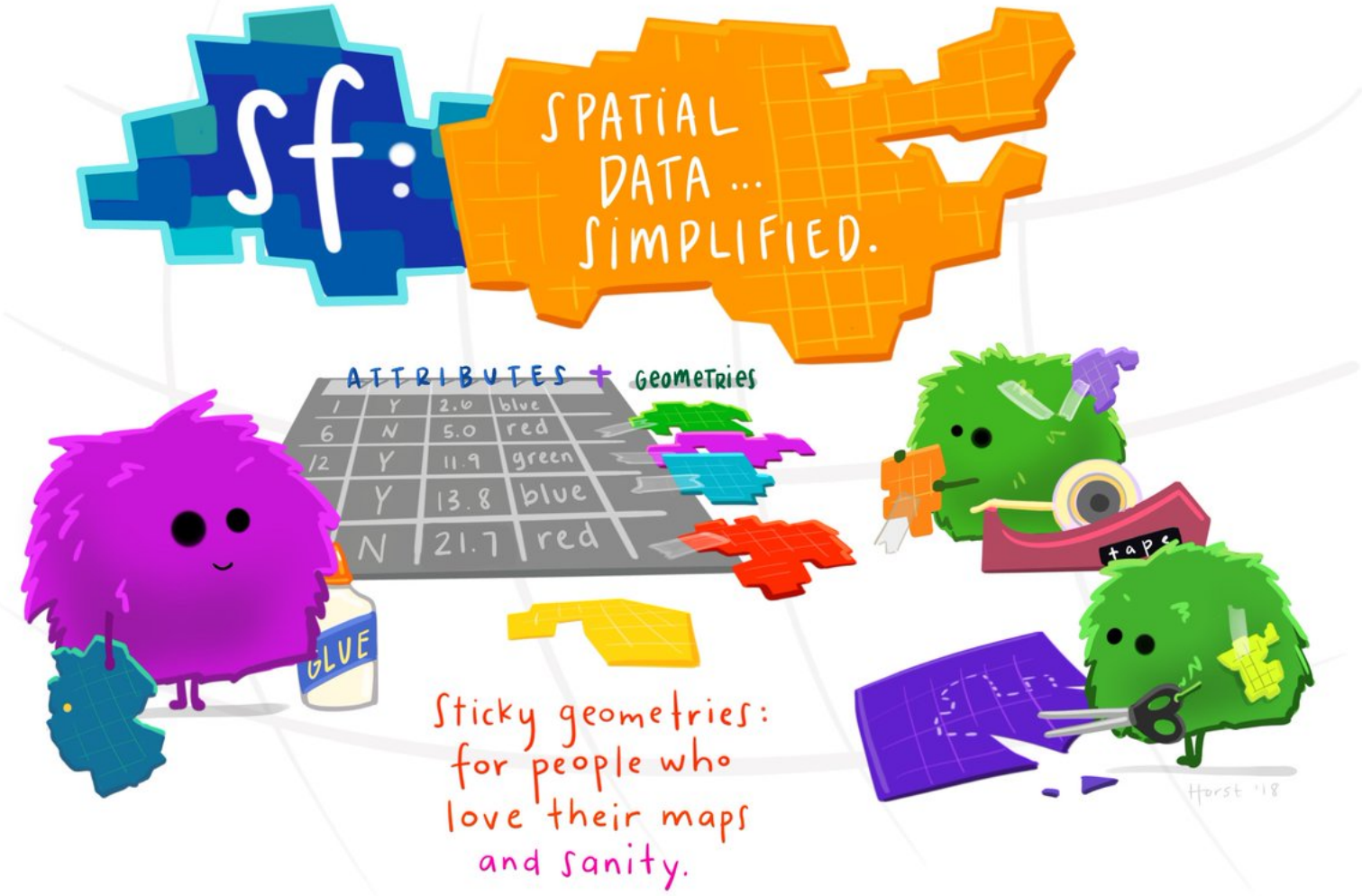


Figure 2: overview

Simple features and data.frames

> *"We usually do not work with geometries of single simple features, but with datasets consisting of sets of features with attributes."*

```
file_name <- system.file("shape/nc.shp", package="sf")

nc <- st_read(file_name)
```

```
## Reading layer `nc' from data source `/Library/Frameworks/R.framework
## Simple feature collection with 100 features and 14 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax:
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
```

Simple features and data.frames

```
head(nc[,c("CNTY_ID", "NAME")])
```

```
## Simple feature collection with 6 features and 2 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -81.74107 ymin: 36.07282 xmax: -75.77316 ymax:
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
##   CNTY_ID        NAME                           geometry
## 1    1825        Ashe MULTIPOLYGON (((-81.47276 3...
## 2    1827   Alleghany MULTIPOLYGON (((-81.23989 3...
## 3    1828       Surry MULTIPOLYGON (((-80.45634 3...
## 4    1831   Currituck MULTIPOLYGON (((-76.00897 3...
## 5    1832 Northampton MULTIPOLYGON (((-77.21767 3...
## 6    1833    Hertford MULTIPOLYGON (((-76.74506 3...
```

Simple features and data.frames

```
## Simple feature collection with 100 features and 6 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -84.32385 ymin: 33.88199 xmax: -75.45698 ymax: 36.58965
## epsg (SRID):    4267
## proj4string:    +proj=longlat +datum=NAD27 +no_defs
## precision:      double (default; no precision model)
## First 3 features:
##   BIR74 SID74 NWBIR74 BIR79 SID79 NWBIR79                              geom
## 1  1091     1      10  1364     0      19 MULTIPOLYGON(((-81.47275543...
## 2   487     0      10   542     3      12 MULTIPOLYGON(((-81.23989105...
## 3  3188     5     208  3616     6     260 MULTIPOLYGON(((-80.45634460...
```

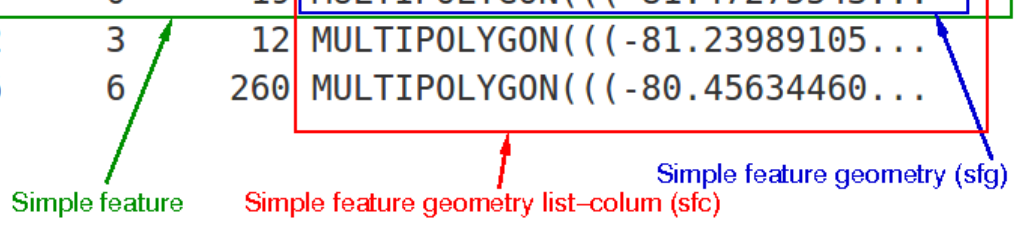


Figure 3: Illustration from Edzer Pebesma

Simple Features and GIS

- Geographic information system
- The package `sf` is very powerfull for building maps.
- Present spatial or geographic data
- It's not (yet) a replacement of ArcGIS of QGIS.

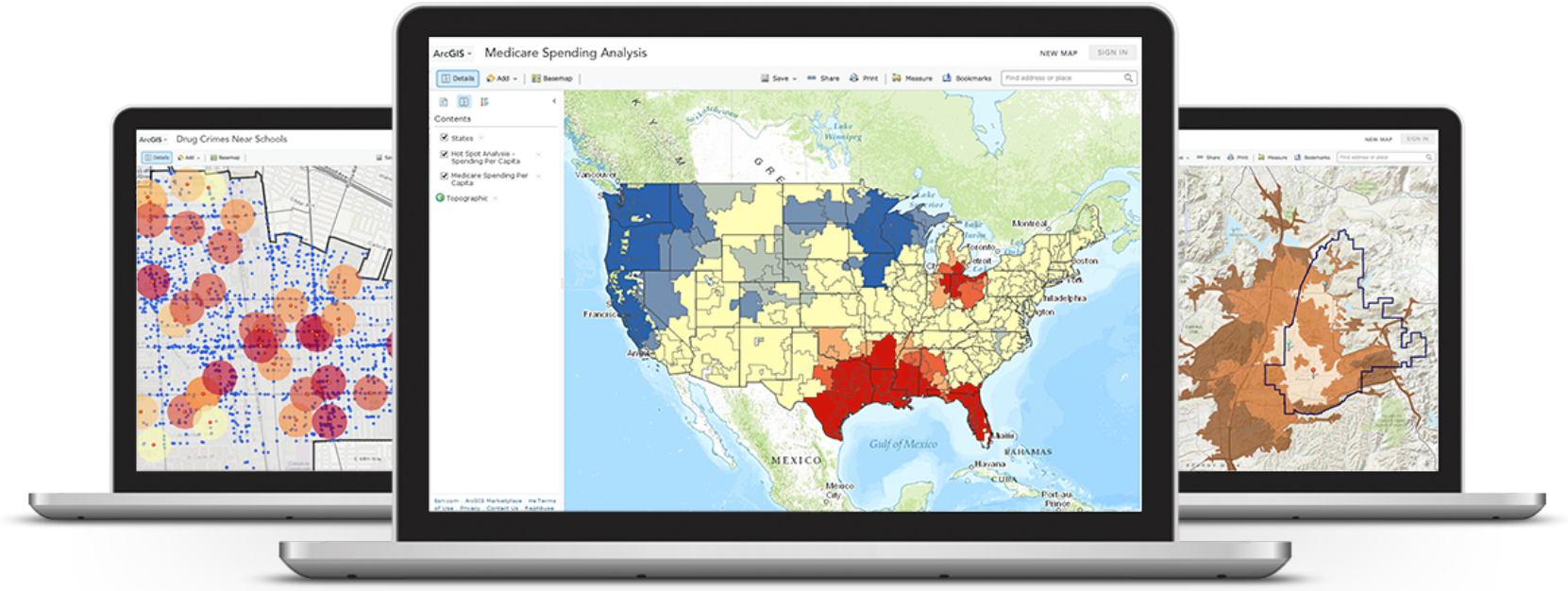


Figure 4: ArcGIS

Example - OpenStreetMap (OSM)

The `osmdata` package helps with extracting data from OpenStreetMap.

```
library("osmdata")
```

Example - OpenStreetMap (OSM)

```
utrecht_sf <- opq(bbox = 'utrecht nl') %>%
  add_osm_feature(key = 'highway', value = 'cycleway') %>%
  osmdata_sf()

utrecht_sf
```

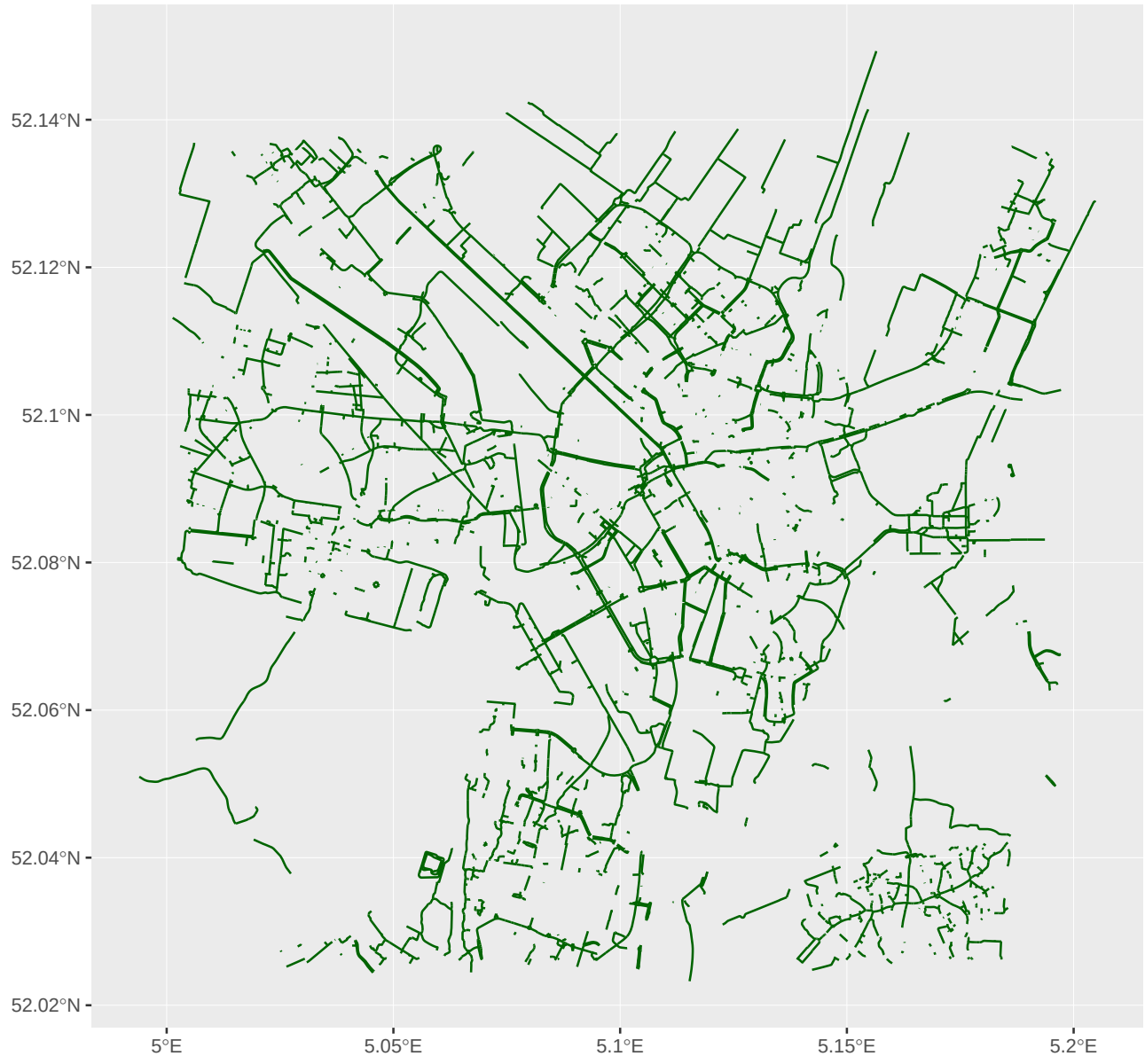
```
## Object of class 'osmdata' with:
##                  $bbox : 52.026282,5.0041608,52.1356715,5.195155
##         $overpass_call : The call submitted to the overpass API
##                 $meta : metadata including timestamp and version nu
##            $osm_points : 'sf' Simple Features Collection with 18876
##             $osm_lines : 'sf' Simple Features Collection with 4445 l
##          $osm_polygons : 'sf' Simple Features Collection with 10 pol
##        $osm_multilines : NULL
##     $osm_multipolygons : NULL
```

Example - OpenStreetMap (OSM)

```
library(ggplot2)

ggplot() +
  geom_sf(
    data=utrecht_sf$osm_lines,
    fill="darkgreen",
    color="darkgreen"
  )
```

Example - OpenStreetMap (OSM)



Example - CBS Dutch municipalities

The osmdata package helps with extracting data from OpenStreetMap.

```
library("cbsshape")
library("dplyr")
```

```
# download 2017 data
wijk_en_buurt_2017 <- st_read_cbs(2017, "data/") # remove "data/"
```

```
## Reading layer `gem_2017' from data source `/Users/jonathandebruin/su
## Simple feature collection with 477 features and 132 fields
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: 10425.16 ymin: 306846.2 xmax: 278026.1 ymax: 6
## epsg (SRID):    NA
## proj4string:    +proj=sterea +lat_0=52.15616055555555 +lon_0=5.38763
```

Example - CBS Dutch municipalities

Extract the geometry of Utrecht

```
sf_cbs_utrecht <- wijk_en_buurt_2017 %>%
  # remove water polygons
  filter(WATER == "NEE", GM_NAAM == "Utrecht") %>%
  st_geometry()
```

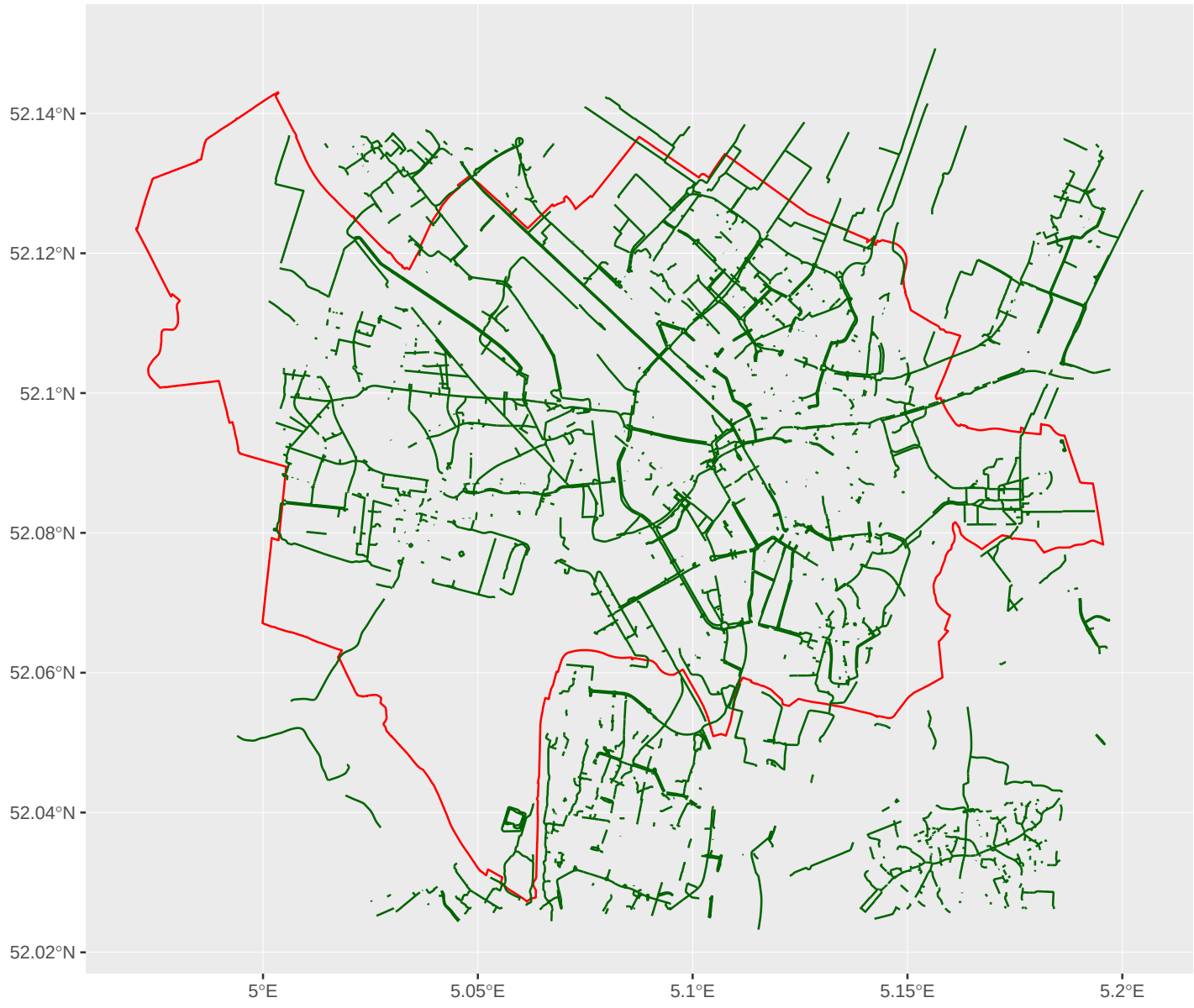
CBS works with RD coordinates (not the typical longitude and latitude)

```
sf_cbs_utrecht <- st_transform(sf_cbs_utrecht, 4326)
# 4326 is wgs84
```

Example - CBS Dutch municipalities

```
ggplot(sf_cbs_utrecht) +
  geom_sf(color="red", alpha=0) +
  geom_sf(
    data=utrecht_sf$osm_lines,
    fill="darkgreen",
    color="darkgreen"
  )
```

Example - CBS Dutch municipalities



Example - ggmap

```
library(ggmap)
```

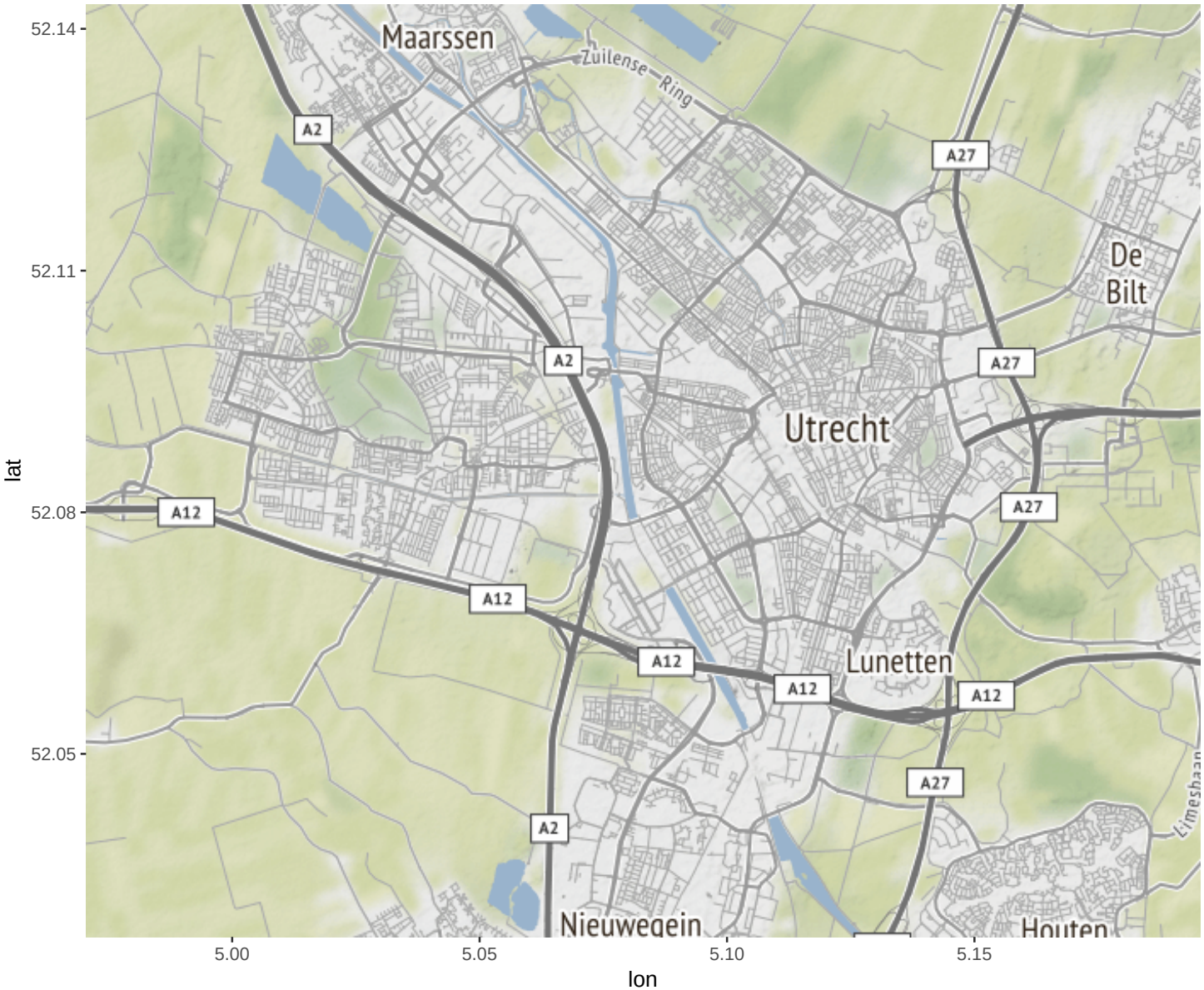
```
st_bbox(sf_cbs_utrecht)
```

```
##       xmin       ymin       xmax       ymax
##   4.970470 52.027255   5.195562 52.143037
```

```
utrecht_map <- get_map(
  c(left = 4.970470,
    bottom = 52.027255,
    right = 5.195562 ,
    top = 52.143037
  ),
  maptype = "toner-background")
```

Example - ggmap

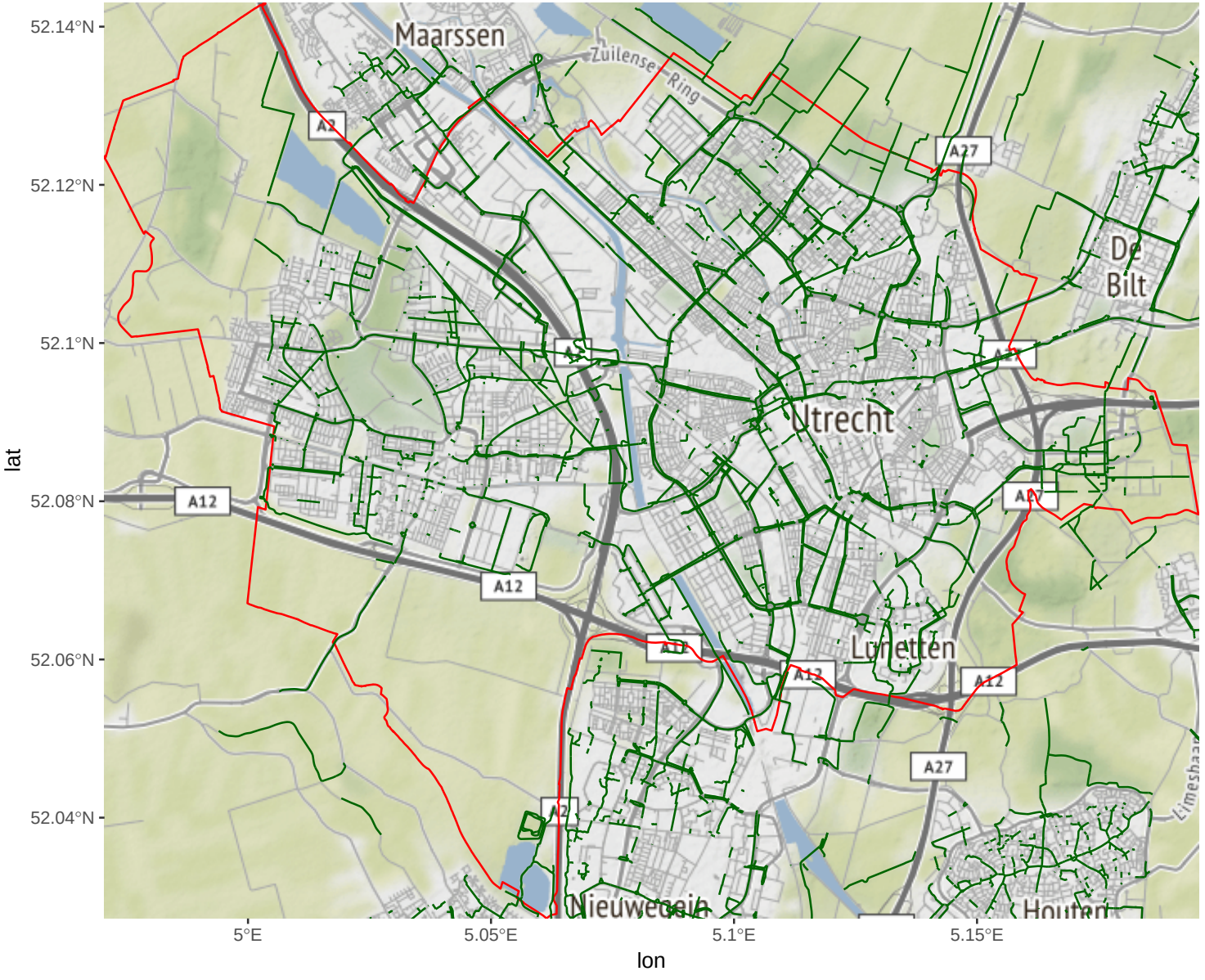
```
ggmap(utrecht_map)
```



Combine OSM, CBS, ggmap

```
ggmap(utrecht_map) +
  geom_sf(data=sf_cbs_utrecht,
          color="red",
          alpha=0,
          inherit.aes =FALSE) +
  geom_sf(
          data=utrecht_sf$osm_lines,
          inherit.aes =FALSE,
          fill="darkgreen",
          color="darkgreen"
  )
```

Combine OSM, CBS, ggmap



Combine OSM, CBS, ggmap - Intersection

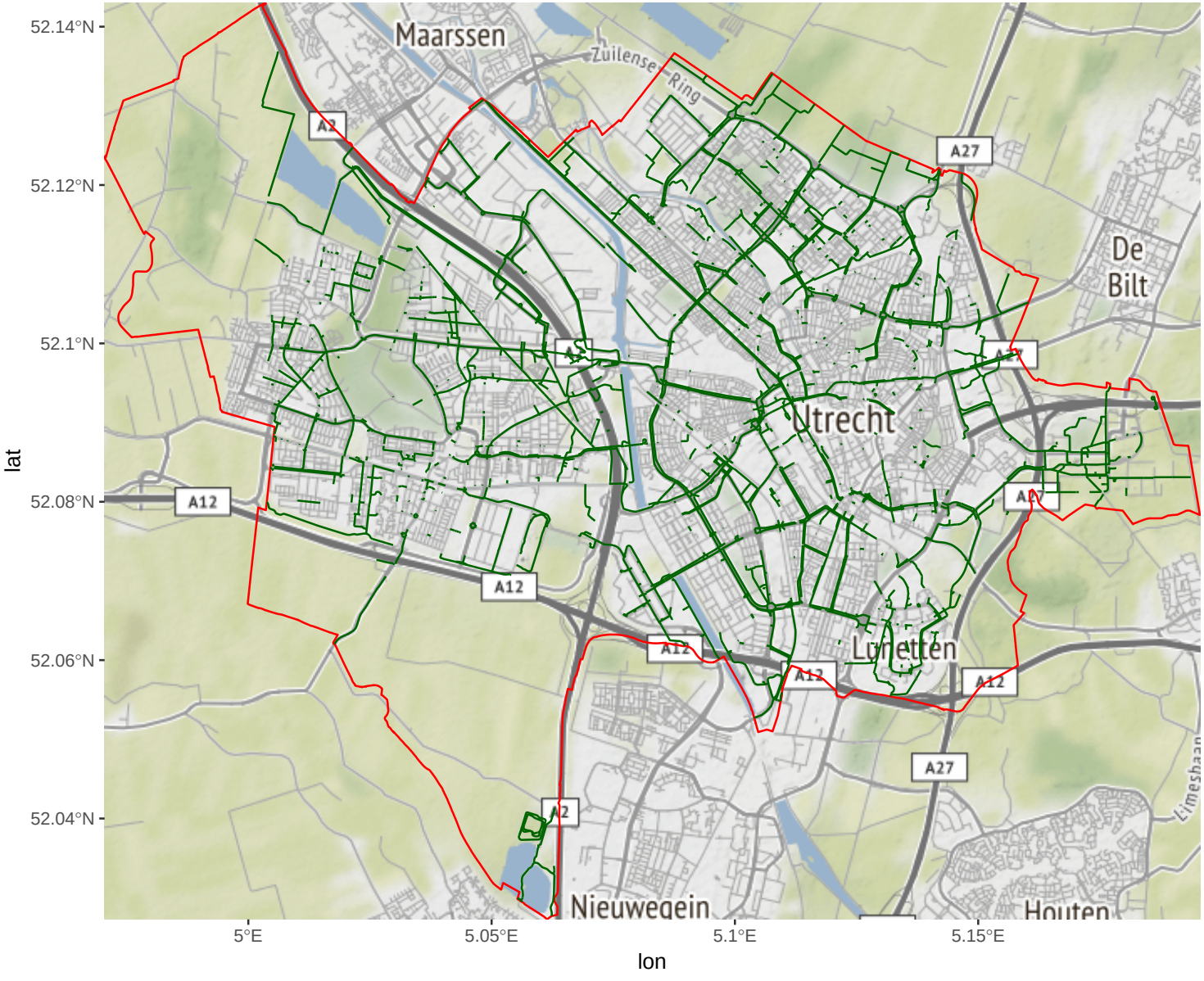
```
(cycle_utrecht <- st_intersection(
  sf_cbs_utrecht,
  utrecht_sf$osm_lines
))
```

```
## Geometry set for 2940 features
## geometry type:  GEOMETRY
## dimension:      XY
## bbox:           xmin: 5.001329 ymin: 52.02767 xmax: 5.19367 ymax: 52
## epsg (SRID):    4326
## proj4string:    +proj=longlat +datum=WGS84 +no_defs
## First 5 geometries:
```

Combine OSM, CBS, ggmap - Plotting

```
ggmap(utrecht_map) +
  geom_sf(data=sf_cbs_utrecht,
          color="red",
          alpha=0,
          inherit.aes =FALSE) +
  geom_sf(
          data=cycle_utrecht,
          inherit.aes =FALSE,
          fill="darkgreen",
          color="darkgreen"
  )
```

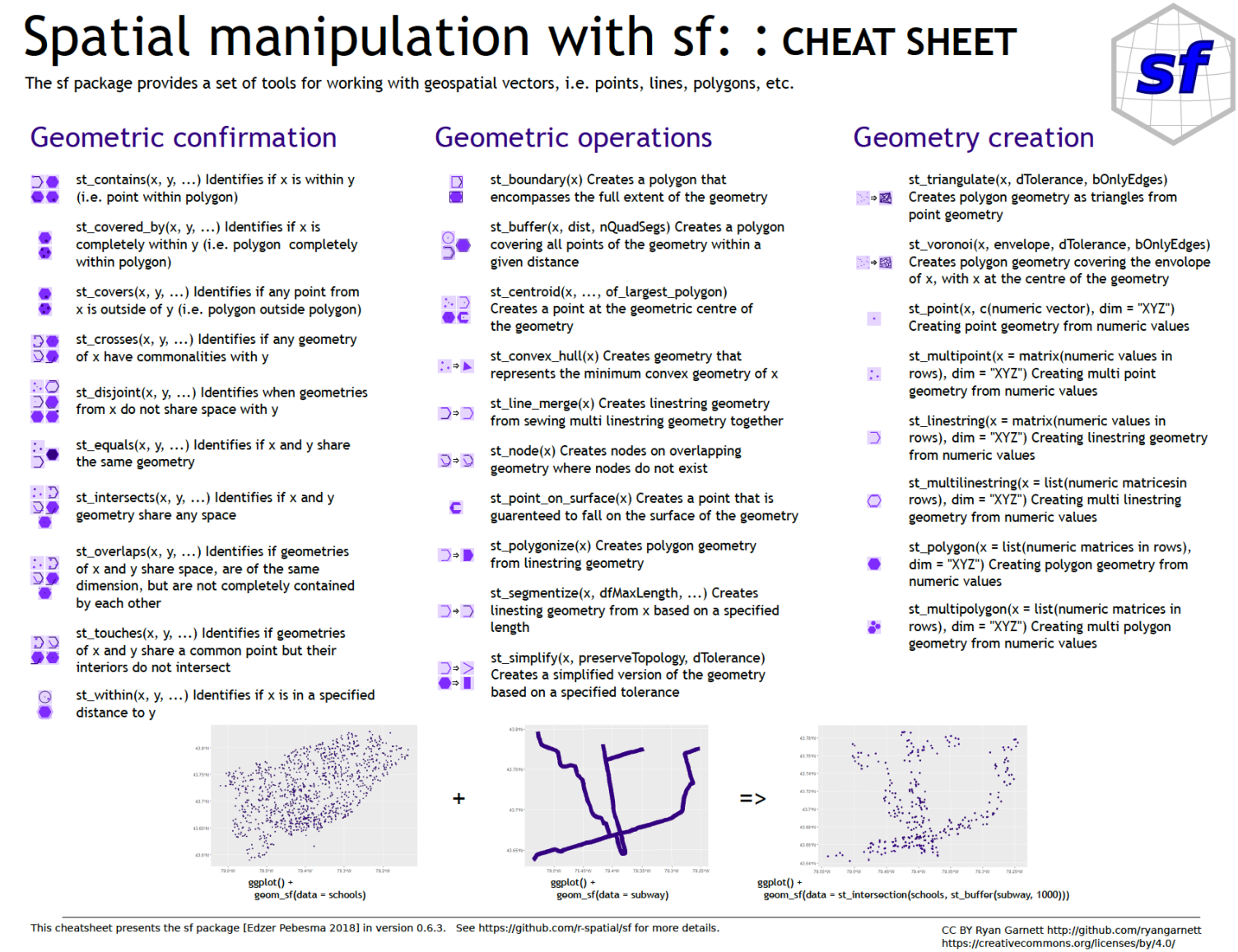
Combine OSM, CBS, ggmap - Result



Resources and examples

1. Simple Features for R
   https://r-spatial.github.io/sf/articles/sf1.html
2. Reading, Writing and Converting Simple Features
   https://r-spatial.github.io/sf/articles/sf2.html
3. Manipulating Simple Feature Geometries
   https://r-spatial.github.io/sf/articles/sf3.html
4. Manipulating Simple Features
   https://r-spatial.github.io/sf/articles/sf4.html
5. Plotting Simple Features
   https://r-spatial.github.io/sf/articles/sf5.html
6. Miscellaneous https://r-spatial.github.io/sf/articles/sf6.html

Resources and examples



Spatial manipulation with sf: : CHEAT SHEET

The sf package provides a set of tools for working with geospatial vectors, i.e. points, lines, polygons, etc.

Geometric confirmation

st_contains(x, y, ...) Identifies if x is within y (i.e. point within polygon)

st_covered_by(x, y, ...) Identifies if x is completely within y (i.e. polygon completely within polygon)

st_covers(x, y, ...) Identifies if any point from x is outside of y (i.e. polygon outside polygon)

st_crosses(x, y, ...) Identifies if any geometry of x have commonalities with y

st_disjoint(x, y, ...) Identifies when geometries from x do not share space with y

st_equals(x, y, ...) Identifies if x and y share the same geometry

st_intersects(x, y, ...) Identifies if x and y geometry share any space

st_overlaps(x, y, ...) Identifies if geometries of x and y share space, are of the same dimension, but are not completely contained by each other

st_touches(x, y, ...) Identifies if geometries of x and y share a common point but their interiors do not intersect

st_within(x, y, ...) Identifies if x is in a specified distance to y

Geometric operations

st_boundary(x) Creates a polygon that encompasses the full extent of the geometry

st_buffer(x, dist, nQuadSegs) Creates a polygon covering all points of the geometry within a given distance

st_centroid(x, ..., of_largest_polygon) Creates a point at the geometric centre of the geometry

st_convex_hull(x) Creates geometry that represents the minimum convex geometry of x

st_line_merge(x) Creates linestring geometry from sewing multi linestring geometry together

st_node(x) Creates nodes on overlapping geometry where nodes do not exist

st_point_on_surface(x) Creates a point that is guarenteed to fall on the surface of the geometry

st_polygonize(x) Creates polygon geometry from linestring geometry

st_segmentize(x, dfMaxLength, ...) Creates linesting geometry from x based on a specified length

st_simplify(x, preserveTopology, dTolerance) Creates a simplified version of the geometry based on a specified tolerance

Geometry creation

st_triangulate(x, dTolerance, bOnlyEdges) Creates polygon geometry as triangles from point geometry

st_voronoi(x, envelope, dTolerance, bOnlyEdges) Creates polygon geometry covering the envolope of x, with x at the centre of the geometry

st_point(x, c(numeric vector), dim = "XYZ") Creating point geometry from numeric values

st_multipoint(x = matrix(numeric values in rows), dim = "XYZ") Creating multi point geometry from numeric values

st_linestring(x = matrix(numeric values in rows), dim = "XYZ") Creating linestring geometry from numeric values

st_multilinestring(x = list(numeric matricesin rows), dim = "XYZ") Creating multi linestring geometry from numeric values

st_polygon(x = list(numeric matrices in rows), dim = "XYZ") Creating polygon geometry from numeric values

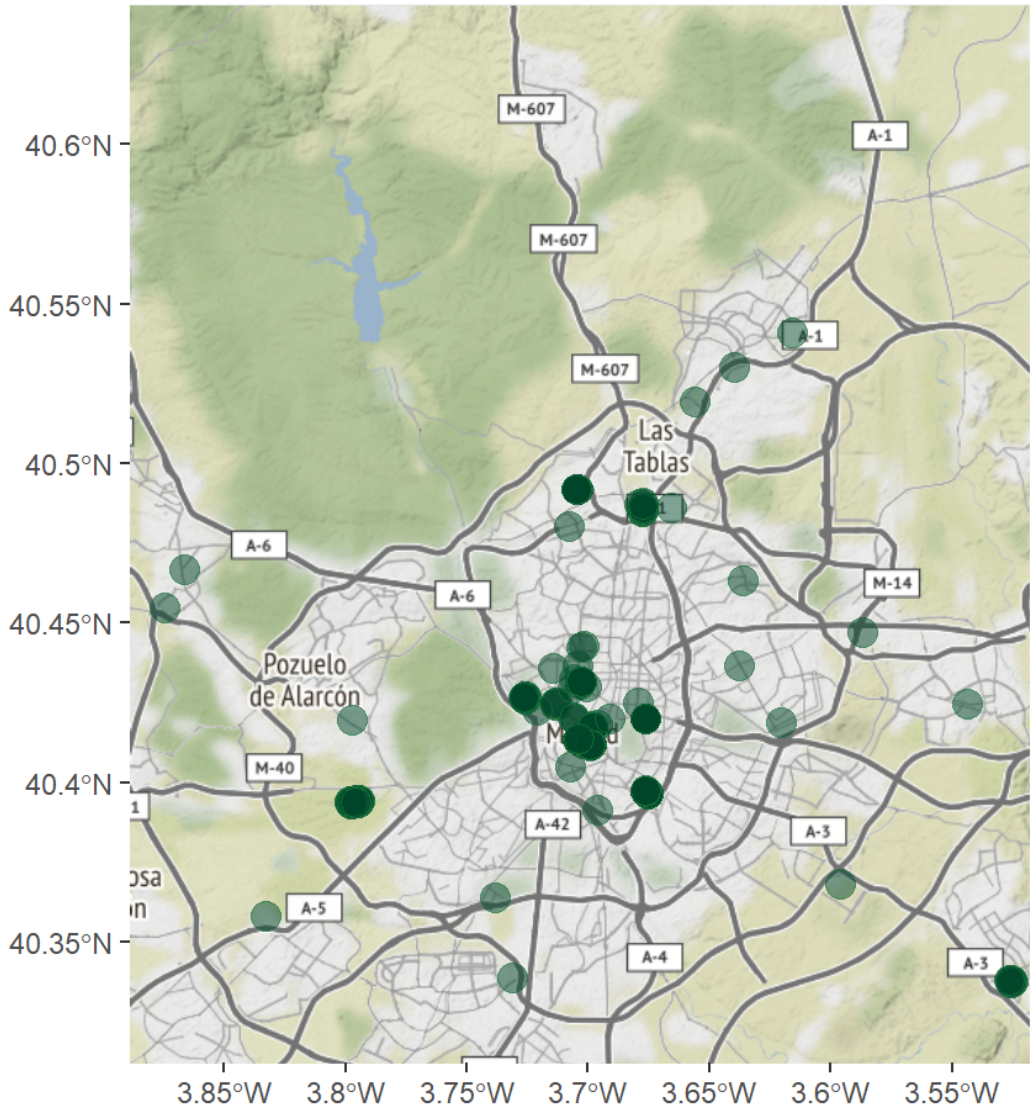
st_multipolygon(x = list(numeric matrices in rows), dim = "XYZ") Creating multi polygon geometry from numeric values

This cheatsheet presents the sf package [Edzer Pebesma 2018] in version 0.6.3. See https://github.com/r-spatial/sf for more details.

Figure 5: Cheatsheet

Resources and examples

https://dominicroye.github.io/en/2018/accessing-openstreetmap-data-with-r/



Questions?

- How to build heatmaps?
- How to enrich data with demographic data of National Statistics?
- How to convert address information into coordinates?
- ...