

# Introdução ao uso de dados geoespaciais no R

## 7 Estrutura e manipulação de dados vetoriais

---

Maurício H. Vancine

Milton C. Ribeiro

UNESP - Rio Claro

Laboratório de Ecologia Espacial e Conservação (LEEC)

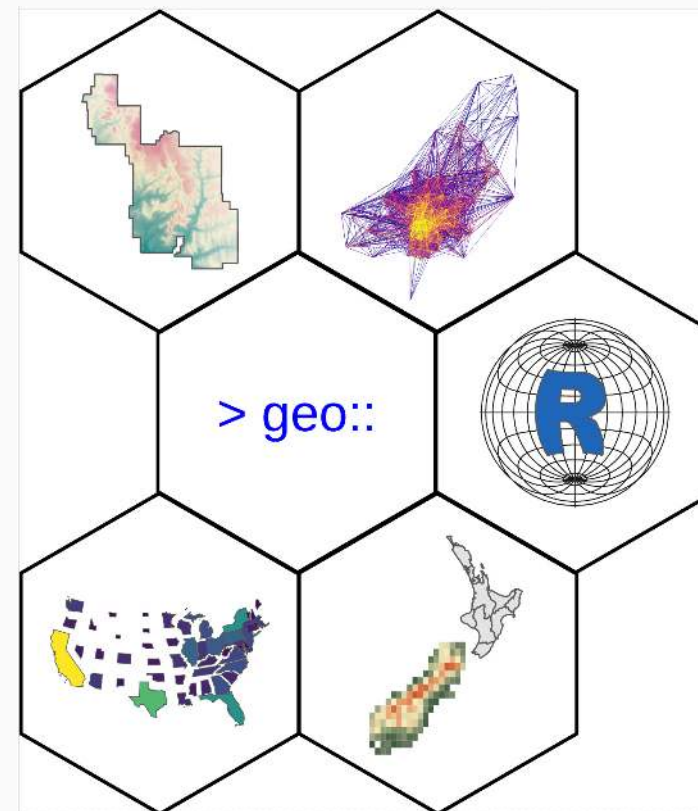
25/10/2021-05/11/2021



# 7 Estrutura e manipulação de vetores

## Tópicos

1. Principais pacotes
2. Geometrias sf
3. Classes sf
4. Importar dados vetoriais
5. Descrição de objetos sf
6. Converter objetos para sf
7. Converter CRS de objetos sf
8. Operações de objetos sf
9. Exportar objetos sf



[Lovelace et al. \(2020\).](#)

# 7 Estrutura e manipulação de vetores

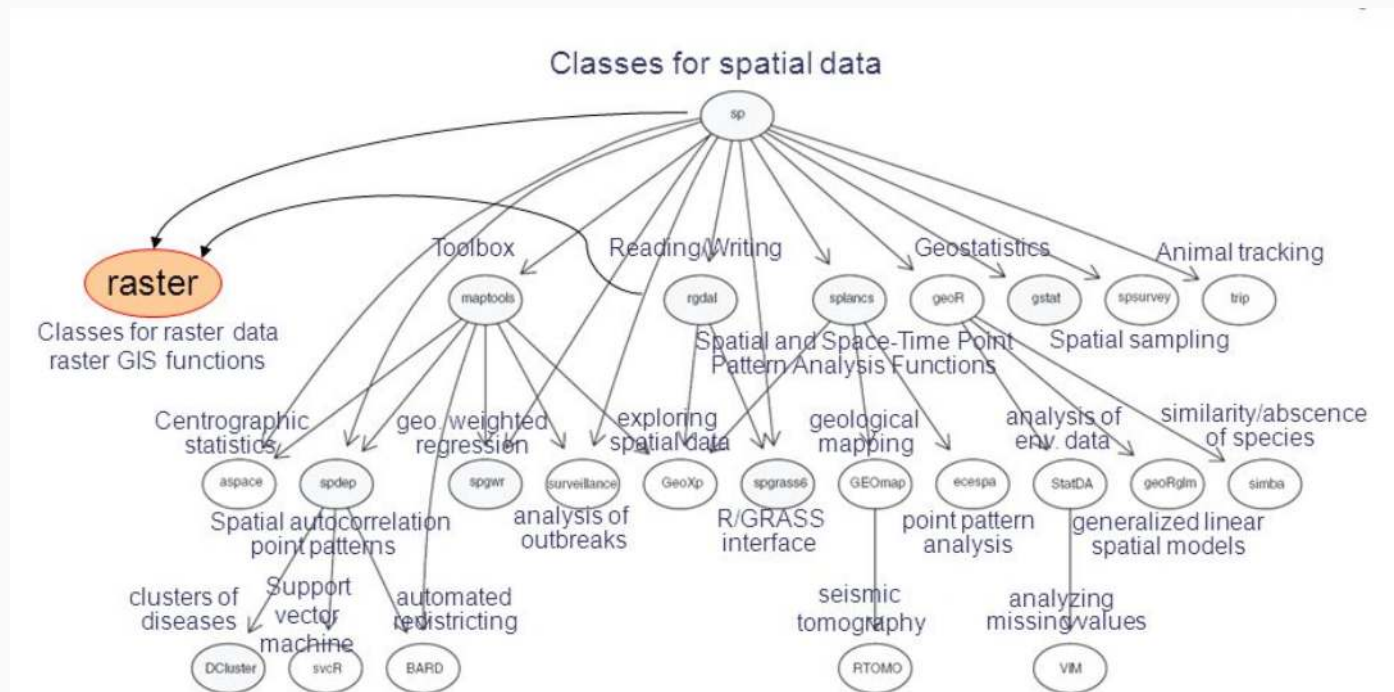
## Script

```
07_script_intro_geoespacial_r.R
```

# 1. Principaux pacotes

## Pacote sp

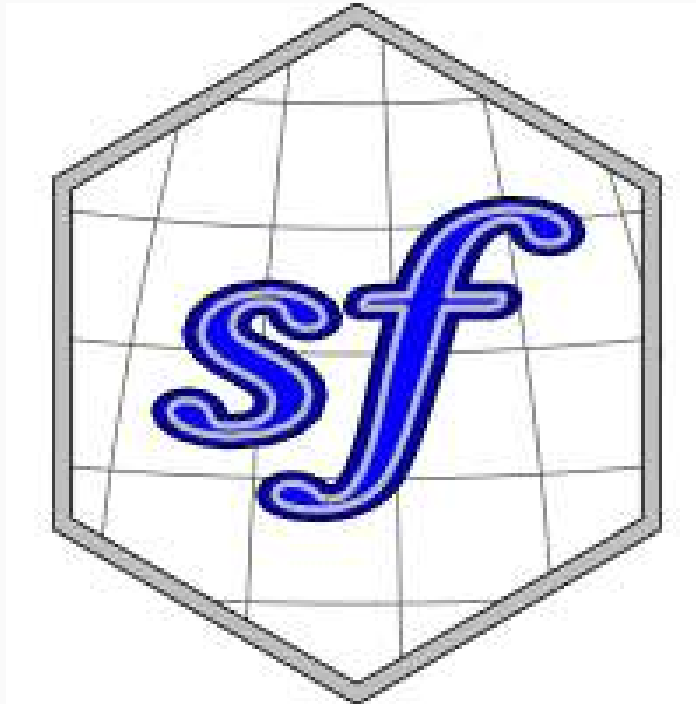
```
# sp  
install.packages("sp")  
library(sp)
```



# 1. Principais pacotes

## Pacote sf

```
# sf  
install.packages("sf")  
library(sf)
```




# 1. Principais pacotes

## Pacote sf

## Spatial manipulation with sf cheat sheets

### 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, nQuadsSegs)` 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 guaranteed to fall on the surface of the geometry
- `st_polygonize(x)` Creates polygon geometry from linestring geometry
- `st_segmentize(x, dfMaxLength, ...)` Creates linestring 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, hOnlyEdges)` Creates polygon geometry covering the envelope of  $x$ , with  $x$  at the centre of the geometry
- `st_point(x, r(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 matrices in 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 cheat sheet presents the `sf` package (first released 2018) in version 0.8.3. See <https://github.com/r-spatial/sf> for more details.

CC BY Ryan Garnett [ryan.garnett@unimelb.edu.au](mailto:ryan.garnett@unimelb.edu.au)  
<https://r-spatial.github.io/sf/cheatsheet/>



# 1. Principais pacotes

## Pacote sf

### Artigo

- Pebesma, Edzer. ["Simple Features for R: Standardized Support for Spatial Vector Data."](#) The R Journal 10.01 (2018): 439-446.

CONTRIBUTED RESEARCH ARTICLE

439

## Simple Features for R: Standardized Support for Spatial Vector Data

by Edzer Pebesma

**Abstract** Simple features are a standardized way of encoding spatial vector data (points, lines, polygons) in computers. The *sf* package implements simple features in R, and has roughly the same capacity for spatial vector data as packages *sp*, *rgeos*, and *rgdal*. We describe the need for this package, its place in the R package ecosystem, and its potential to connect R to other computer systems. We illustrate this with examples of its use.

### What are simple features?

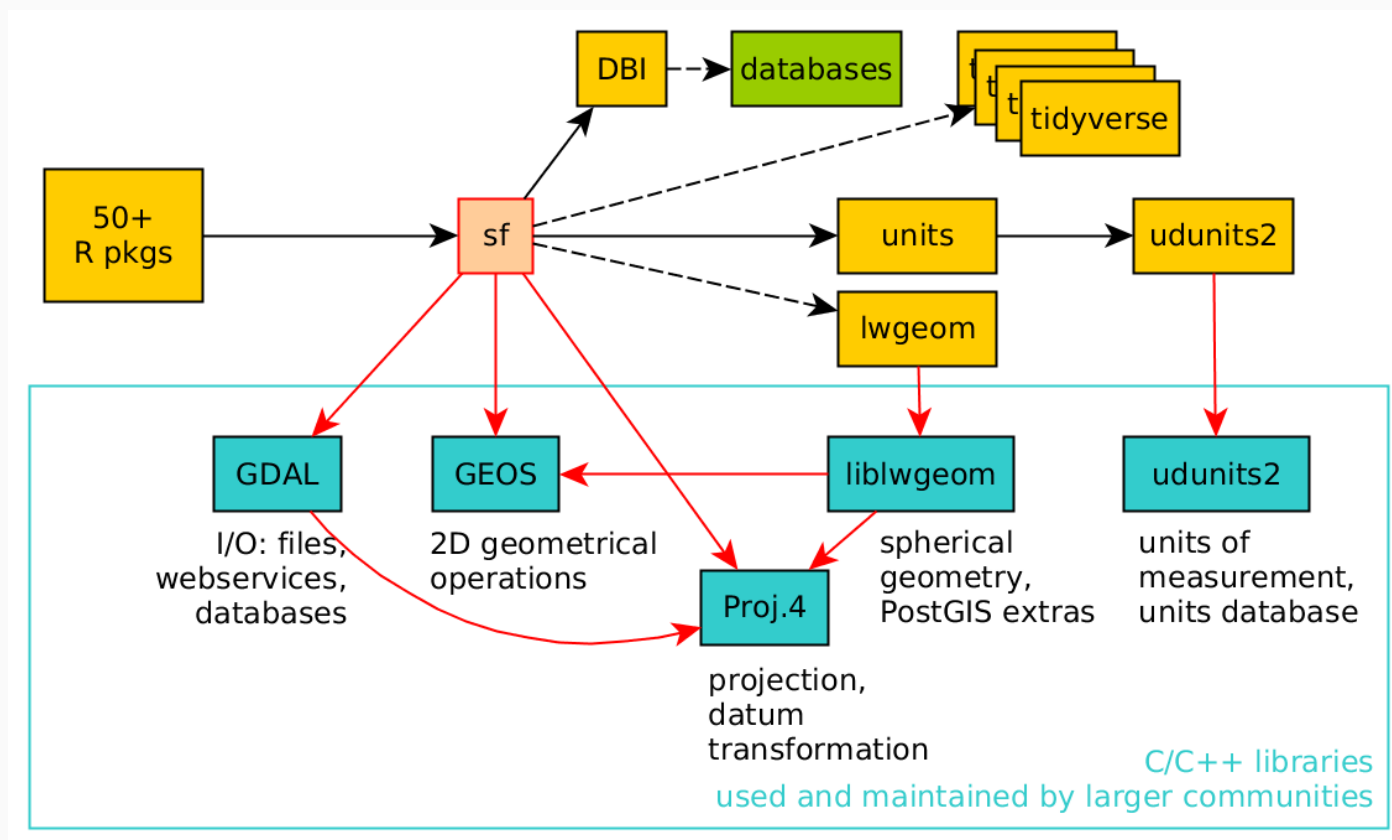
Features can be thought of as “things” or objects that have a spatial location or extent; they may be physical objects like a building, or social conventions like a political state. *Feature geometry* refers to the spatial properties (location or extent) of a feature, and can be described by a point, a point set, a linestring, a set of linestrings, a polygon, a set of polygons, or a combination of these. The *simple* adjective of *simple features* refers to the property that linestrings and polygons are built from points connected by *straight* line segments. Features typically also have other properties (temporal properties, color, name, measured quantity), which are called *feature attributes*. Not all spatial phenomena are easy to represent by “things or objects:” continuous phenomena such as water temperature or elevation are better represented as *functions* mapping from continuous or sampled space (and time) to values (Scheider et al., 2016), and are often represented by *raster* data rather than vector (points, lines, polygons) data.



# 1. Principais pacotes

## Pacote sf

### Dependências de outros pacotes



# 1. Principais pacotes

## Pacote sf

## Métodos

class	methods
sfg	as.matrix, c, coerce, format, head, Ops, plot, print, st_as_binary, st_as_grob, st_as_text, st_transform, st_coordinates, st_geometry, st_boundary, st_buffer, st_centroid, st_convex_hull, st_difference, st_intersection, st_line_merge, st_make_valid, st_node, st_point_on_surface, st_polygonize, st_segmentize, st_simplify, st_split, st_sym_difference, st_triangulate, st_union, st_voronoi, st_cast, st_collection_extract, st_is, st_zm
sfc	[, [<-, as.data.frame, c, coerce, format, Ops, print, rep, st_as_binary, st_as_text, st_bbox, st_coordinates, st_crs, st_crs<-, st_geometry, st_precision, st_set_precision, str, summary, st_boundary, st_buffer, st_centroid, st_convex_hull, st_difference, st_intersection, st_line_merge, st_make_valid, st_node, st_point_on_surface, st_polygonize, st_segmentize, st_simplify, st_split, st_sym_difference, st_transform, st_triangulate, st_union, st_voronoi, st_cast, st_collection_extract, st_is, st_zm, obj_sum, type_sum
sf	[, [[<-, \$<-, aggregate, cbind, coerce, merge, plot, print, rbind, st_agr, st_agr<-, st_bbox, st_coordinates, st_crs, st_crs<-, st_geometry, st_geometry<-, st_precision, st_set_precision, st_transform, st_boundary, st_buffer, st_centroid, st_convex_hull, st_difference, st_intersection, st_line_merge, st_make_valid, st_node, st_point_on_surface, st_polygonize, st_segmentize, st_simplify, st_split, st_sym_difference, st_triangulate, st_union, st_voronoi, st_cast, st_collection_extract, st_is, st_zm, anti_join, arrange, distinct, filter, full_join, gather, group_by, inner_join, left_join, nest, mutate, rename, right_join, sample_frac, sample_n, select, semi_join, separate, slice, spread, summarise, transmute, ungroup, unite
crs	\$, is.na, Ops, print, st_as_text, st_crs

# 1. Principais pacotes

## Pacote sf

### Funções

category	functions
binary predicates	st_contains, st_contains_properly, st_covered_by, st_covers, st_crosses, st_disjoint, st_equals, st_equals_exact, st_intersects, st_is_within_distance, st_within, st_touches, st_overlaps
binary operations	st_relate, st_distance
unary operations	st_dimension, st_area, st_length, st_is_longlat, st_is_simple, st_is_valid, st_jitter, st_geohash, st_geometry_type
miscellaneous	st_sample, st_line_sample, st_join, st_interpolate_aw, st_make_grid, st_graticule, sf_extSoftVersion, rawToHex, st_proj_info
setters	st_set_agr, st_set_crs
constructors	st_sfc, st_sf, st_as_sf, st_as_sfc, st_point, st_multipoint, st_linestring, st_multilinestring, st_polygon, st_multipolygon, st_geometrycollection, st_combine, st_bind_cols
in- & output	st_read, st_read_db, st_write, st_write_db, read_sf, write_sf, st_drivers, st_layers
plotting	st_viewport, st_wrap_dateline, sf.colors

# 1. Principais pacotes

Pacote sf - Descrição

**Rápida importação e exportação** de dados

**Desempenho aprimorado** de mapas e gráficos







Objetos `sf` podem ser tratados como `data frames` (`tibbles`) na maioria das **operações de manipulação**

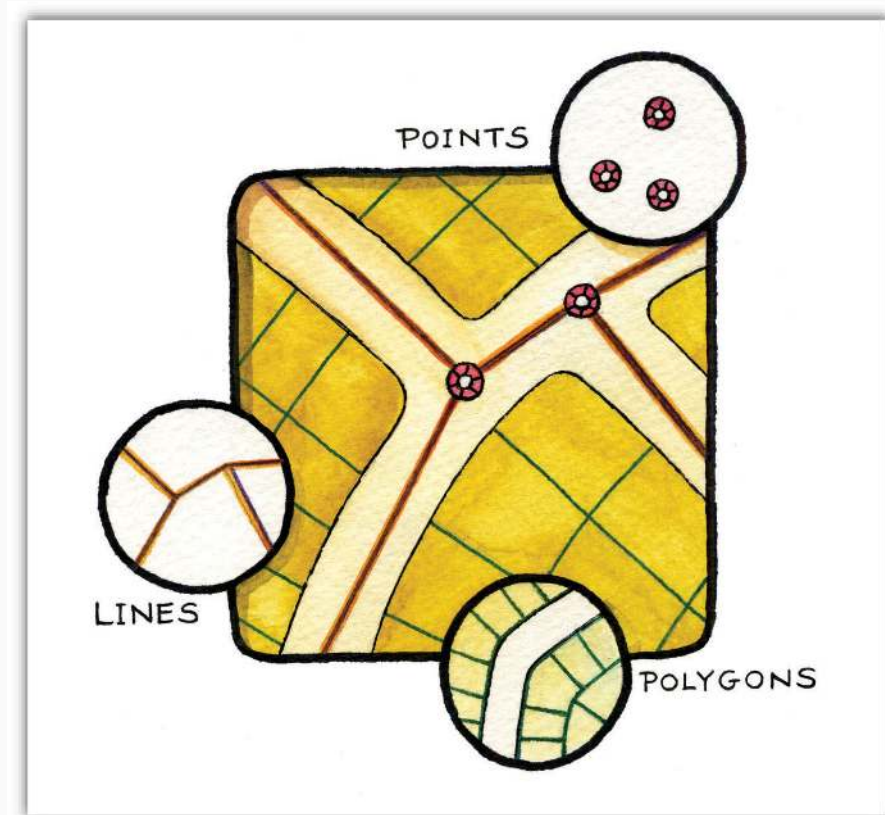
As funções `sf` **podem ser combinadas** usando o operador `%>%` e funcionam bem com `tidyverse`

Os nomes das funções sf são relativamente **consistentes e intuitivos** (todos começam com `sf::st_`)

## 2. Tipos de geometrias sf

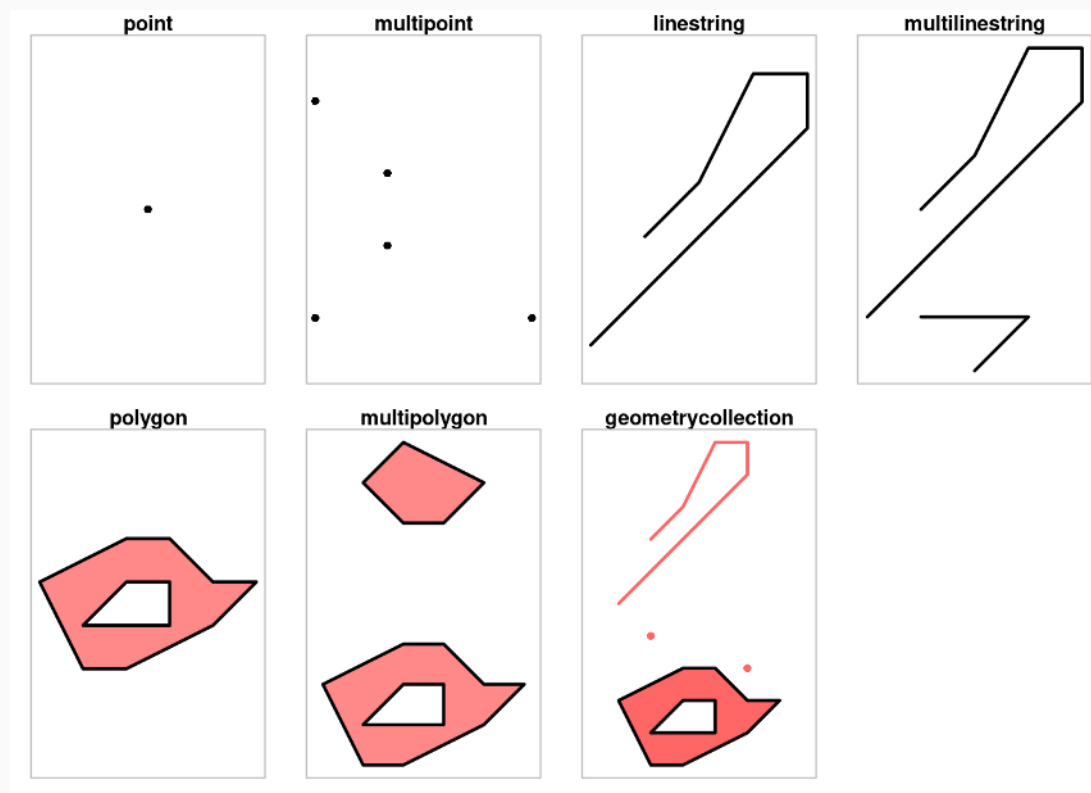
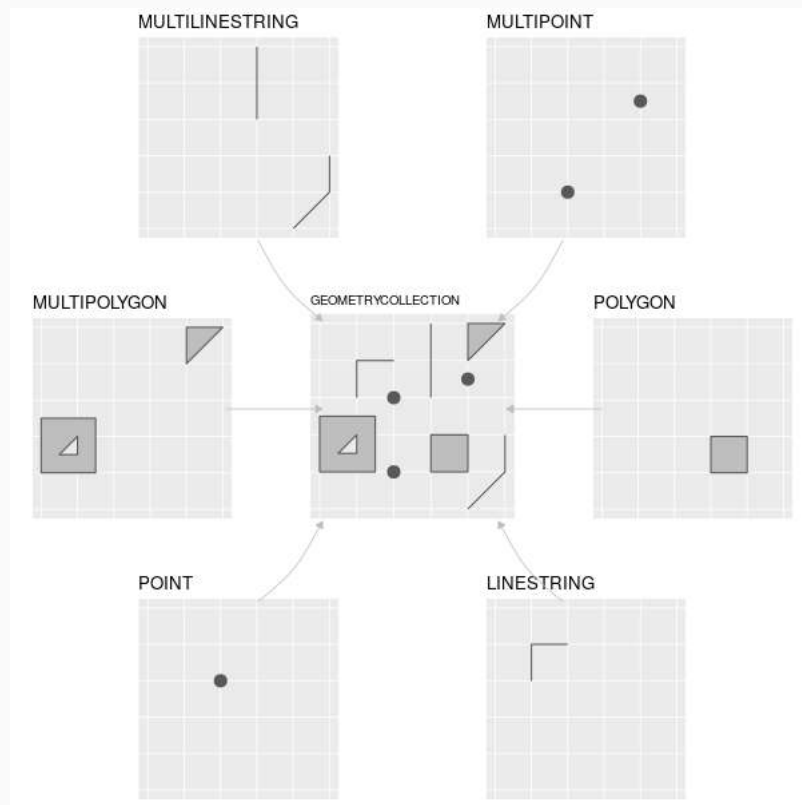
### Tipos de dados geoespaciais vetoriais

Geometrias	Entidade espacial	Representação	Atributos																				
Pontos			<table><thead><tr><th>FID</th><th>Município</th><th>Hidrografia</th><th>Vazão</th></tr></thead><tbody><tr><td>1</td><td>Rio Claro</td><td>Nascente</td><td>0,2</td></tr><tr><td>2</td><td>Rio Claro</td><td>Nascente</td><td>0,8</td></tr><tr><td>3</td><td>Rio Claro</td><td>Nascente</td><td>1,1</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td></tr></tbody></table>	FID	Município	Hidrografia	Vazão	1	Rio Claro	Nascente	0,2	2	Rio Claro	Nascente	0,8	3	Rio Claro	Nascente	1,1	...	...	...	...
FID	Município	Hidrografia	Vazão																				
1	Rio Claro	Nascente	0,2																				
2	Rio Claro	Nascente	0,8																				
3	Rio Claro	Nascente	1,1																				
...	...	...	...																				
Linhas			<table><thead><tr><th>FID</th><th>Município</th><th>Hidrografia</th><th>Vazão</th></tr></thead><tbody><tr><td>1</td><td>Rio Claro</td><td>Rios</td><td>2,4</td></tr><tr><td>2</td><td>Rio Claro</td><td>Rios</td><td>3,1</td></tr><tr><td>3</td><td>Rio Claro</td><td>Rios</td><td>7,7</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td></tr></tbody></table>	FID	Município	Hidrografia	Vazão	1	Rio Claro	Rios	2,4	2	Rio Claro	Rios	3,1	3	Rio Claro	Rios	7,7	...	...	...	...
FID	Município	Hidrografia	Vazão																				
1	Rio Claro	Rios	2,4																				
2	Rio Claro	Rios	3,1																				
3	Rio Claro	Rios	7,7																				
...	...	...	...																				
Polígonos			<table><thead><tr><th>FID</th><th>Município</th><th>Uso</th><th>Área</th></tr></thead><tbody><tr><td>1</td><td>Rio Claro</td><td>Floresta</td><td>10,1</td></tr><tr><td>2</td><td>Rio Claro</td><td>Floresta</td><td>19,8</td></tr><tr><td>3</td><td>Rio Claro</td><td>Floresta</td><td>50,2</td></tr><tr><td>...</td><td>...</td><td>...</td><td>...</td></tr></tbody></table>	FID	Município	Uso	Área	1	Rio Claro	Floresta	10,1	2	Rio Claro	Floresta	19,8	3	Rio Claro	Floresta	50,2	...	...	...	...
FID	Município	Uso	Área																				
1	Rio Claro	Floresta	10,1																				
2	Rio Claro	Floresta	19,8																				
3	Rio Claro	Floresta	50,2																				
...	...	...	...																				



## 2. Tipos de geometrias sf

### Tipos de geometrias



# 3. Classes sf

O pacote sf define um sistema de três classes hierárquicas

Classes	Hierarquia	Informação
1. Simple feature geometries ( <b>g</b> )	Geometria	Tipo de geometria e coordenadas
2. Simple feature columns ( <b>c</b> )	Coluna de geometria	Conjunto de <b>g</b> + CRS
3. Simple feature ()	Camada	Conjunto de <b>c</b> + atributos

- 1. Classe `sfg` - uma geometria única
- 1. Classe `sfc` - uma coluna de geometria, que é um conjunto de geometrias `sfg` e informações Sistema de Referência de Coordenadas (do inglês *Coordinate Reference Systems* - CRS)
- 1. Classe `sf` - uma camada, que é uma coluna de geometria `sfc` dentro de um data frame com atributos não espaciais (tabela de atributos)



# 3. Classes sf

## 1. Simple feature geometries (sfg)

A classe `sfg` representa os **diferentes tipos de geometrias** no R: ponto, linha, polígono (e seus equivalentes 'multi') ou coleção de geometrias

Funções para criar geometrias da classe `sfg`:

```
# simple
sf::st_point()
sf::st_linestring()
sf::st_polygon()

# multi
sf::st_multipoint()
sf::st_multilinestring()
sf::st_multipolygon()

# collections
sf::st_geometrycollection()
```

# 3. Classes sf

## 1. Simple feature geometries (`sfg`)

Os objetos `sfg` podem ser **criados** a partir de **três tipos de dados R base**:

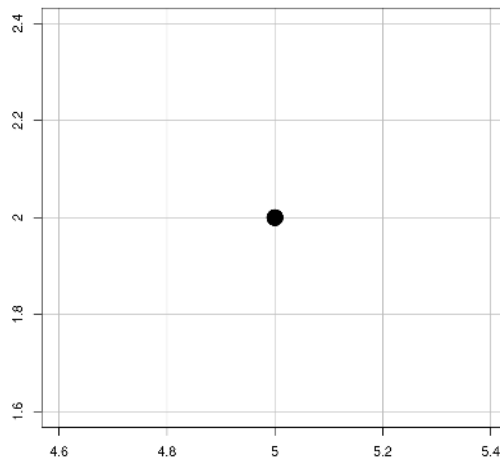
1. **vetor numérico**: um único ponto
2. **matriz**: um conjunto de pontos, onde cada linha representa um ponto, um multiponto ou uma linha
3. **lista**: uma coleção de objetos como matrizes, cadeias multilinha ou coleções de geometrias

# 3. Classes sf

## 1. Simple feature geometries (sfg)

```
# vector - point  
vec ← c(5, 2)  
po ← sf::st_point(vec)  
po
```

```
# plot  
plot(po, pch = 20, cex = 4, axes = TRUE, graticule = TRUE)
```

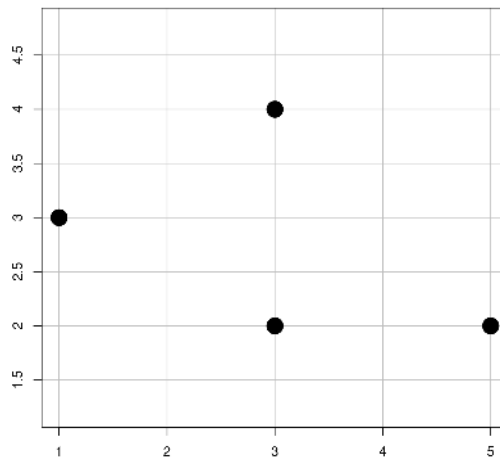


# 3. Classes sf

## 1. Simple feature geometries (sfg)

```
# matrix - multipoint
multipoint_matrix <- rbind(c(5, 2), c(1, 3), c(3, 4), c(3, 2))
po_mul <- sf::st_multipoint(multipoint_matrix)
po_mul
```

```
# plot
plot(po_mul, pch = 20, cex = 4, axes = TRUE, graticule = TRUE)
```



# 3. Classes sf

## 1. Simple feature geometries (sfg)

```
# matrix - linestring  
multipoint_matrix ← rbind(c(5, 2), c(1, 3), c(3, 4), c(3, 2))  
lin ← sf::st_linestring(multipoint_matrix)  
lin
```

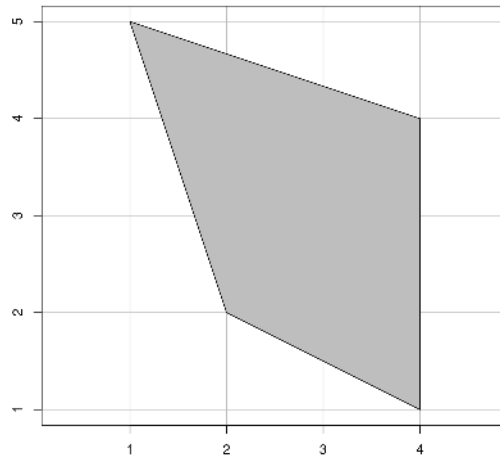
```
# plot  
plot(lin, lwd = 2, axes = TRUE, graticule = TRUE)
```

# 3. Classes sf

## 1. Simple feature geometries (sfg)

```
# list - polygon
polygon_list ← list(rbind(c(1, 5), c(2, 2), c(4, 1), c(4, 4), c(1, 5)))
pol ← sf::st_polygon(polygon_list)
pol
```

```
# plot
plot(pol, col = "gray", axes = TRUE, graticule = TRUE)
```



# 3. Classes sf

## 2. Simple feature columns (`sfc`)

Uma **lista de objetos** `sfg` que possui o **mesmo CRS** (*Coordinate Reference System*)

Pode conter objetos da **mesma geometria** ou de **geometrias diferentes**

Funções para **criar** a classe `sfc` e verificar o **tipo da geometria e CRS**:

```
sf::st_sfc()  
sf::st_geometry_type()  
sf::st_crs()
```



# 3. Classes sf

## 2. Simple feature columns (sfc)

```
# sfc point
point1 <- sf::st_point(c(5, 2))
point2 <- sf::st_point(c(1, 3))
points_sfc <- sf::st_sfc(point1, point2)
points_sfc
```

```
## Geometry set for 2 features
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: 1 ymin: 2 xmax: 5 ymax: 3
## CRS:            NA
```

```
sf::st_geometry_type(points_sfc)
```

```
## [1] POINT POINT
```

```
## 18 Levels: GEOMETRY POINT LINESTRING POLYGON MULTIPOINT MULTILINESTRING MULTIPOLYGON GEOMETRYCOLLECTION CIRCULARST
```

# 3. Classes sf

## 2. Simple feature columns (sfc)

```
# plot  
plot(points_sfc, pch = 20, cex = 4, axes = TRUE, graticule = TRUE)
```

# 3. Classes sf

## 2. Simple feature columns (sfc)

```
# sfc geometry
po_pol_sfc <- sf::st_sfc(po, pol)
po_pol_sfc
```

```
## Geometry set for 2 features
## Geometry type: GEOMETRY
## Dimension:      XY
## Bounding box:   xmin: 1 ymin: 1 xmax: 5 ymax: 5
## CRS:            NA
```

```
sf::st_geometry_type(po_pol_sfc)
```

```
## [1] POINT POLYGON
## 18 Levels: GEOMETRY POINT LINESTRING POLYGON MULTIPOINT MULTILINESTRING MULTIPOLYGON GEOMETRYCOLLECTION CIRCULARST
```

# 3. Classes sf

## 2. Simple feature columns (`sfc`)

```
# plot  
plot(po_pol_sfc, pch = 20, cex = 4, lwd = 2, col = "gray", axes = TRUE, graticule = TRUE)
```

# 3. Classes sf

## 2. Simple feature columns (sf)

### Coordinate Reference Systems (CRS) - EPSG

```
# epsg definition
points_sfcrs_wgs <- sf::st_sf(point1, point2, crs = 4326)
sf::st_crs(points_sfcrs_wgs)
```

```
## Coordinate Reference System:
##   User input: EPSG:4326
##   wkt:
##   GEOGCRS["WGS 84",
##     DATUM["World Geodetic System 1984",
##       ELLIPSOID["WGS 84",6378137,298.257223563,
##         LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##       ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##       AXIS["geodetic latitude (Lat)",north,
##         ORDER[1],
##         ANGLEUNIT["degree",0.0174532925199433]],
##       AXIS["geodetic longitude (Lon)",east,
```

# 3. Classes sf

## 2. Simple feature columns (sf)

### Coordinate Reference Systems (CRS) - proj4string

```
# proj4string definition
points_sfcrs_wgs <- sf::st_sf(point1, point2, crs = "+proj=longlat +datum=WGS84 +no_defs")
sf::st_crs(points_sfcrs_wgs)
```

```
## Coordinate Reference System:
##   User input: +proj=longlat +datum=WGS84 +no_defs
##   wkt:
##   GEOGCRS["unknown",
##     DATUM["World Geodetic System 1984",
##       ELLIPSOID["WGS 84",6378137,298.257223563,
##         LENGTHUNIT["metre",1]],
##       ID["EPSG",6326]],
##     PRIMEM["Greenwich",0,
##       ANGLEUNIT["degree",0.0174532925199433],
##       ID["EPSG",8901]],
##     CS[ellipsoidal,2],
##     AXIS["longitude",east,
##       ORDER[1],
```

# 3. Classes sf

## 2. Simple feature columns (sfc)

```
# plot  
plot(points_sfc_wgs, pch = 20, cex = 4, axes = TRUE, graticule = TRUE)
```



## 3. Classes sf

### 3. Simple feature (sf)

A classe sf (simple features) são *data frames* com **atributos espaciais** armazenados em uma coluna, geralmente chamada de **geometry**

Essa **dualidade** é central para o conceito de **simple features**: na maioria das vezes, um sf pode **ser tratado e se comporta** como um *data frame*

**Simple features** são, em essência, *data frames* com uma **extensão espacial**

# 3. Classes sf

## 3. Simple feature (sf)

Criar um objeto sf

```
rc_point ← sf::st_point(c(-47.57, -22.39))           # `sfg` object
rc_geom ← sf::st_sfc(rc_point, crs = 4326)           # `sfc` object
rc_attrib ← data.frame(                             # data frame object
  name = "Rio Claro",
  temperature = 19,
  date = as.Date("2020-10-13")
)
rc_sf ← sf::st_sf(rc_attrib, geometry = rc_geom)     # sf object
```

# 3. Classes sf

## 3. Simple feature (sf)

### A estrutura de um objeto sf

```
rc_sf
```

```
## Simple feature collection with 1 feature and 3 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: -47.57 ymin: -22.39 xmax: -47.57 ymax: -22.39
## Geodetic CRS:   WGS 84
##      name temperature      date      geometry
## 1 Rio Claro      19 2020-10-13 POINT (-47.57 -22.39)
```

### As duas classes do objeto sf

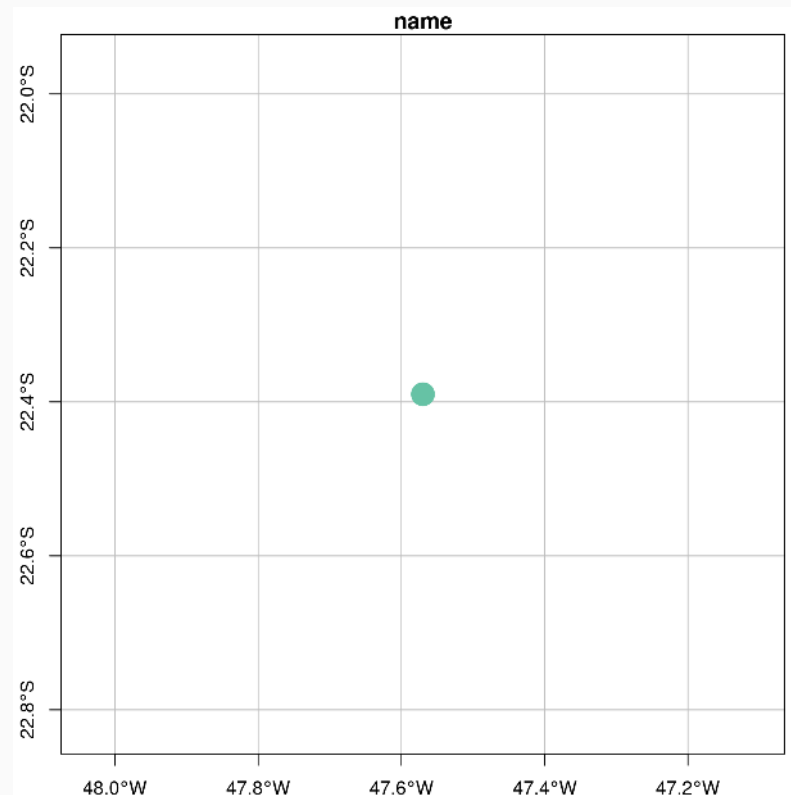
```
class(rc_sf)
```

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

# 3. Classes sf

## 3. Simple feature (sf)

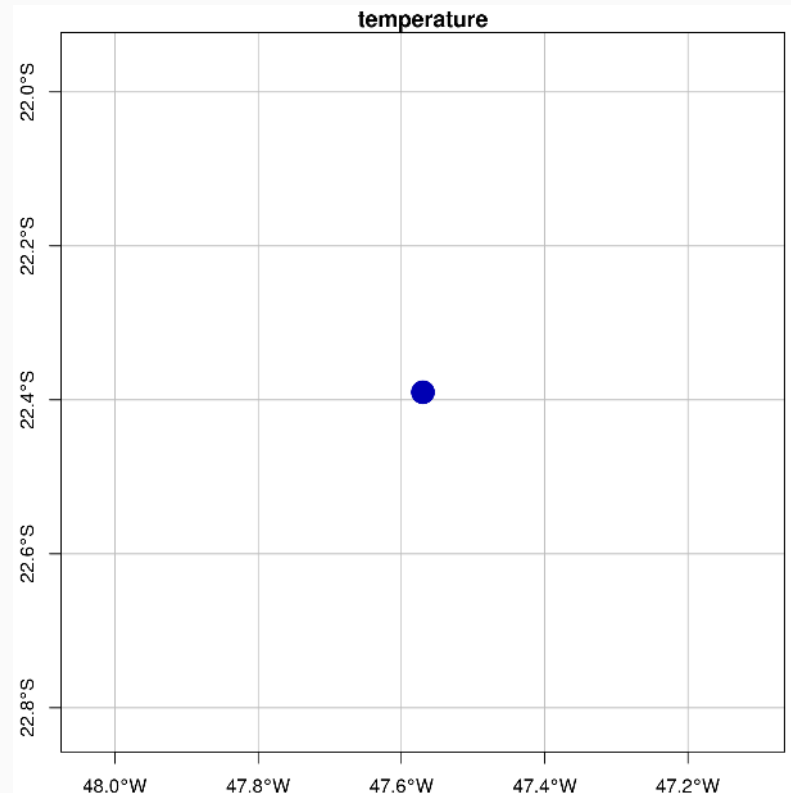
```
# plot  
plot(rc_sf[1], pch = 20, cex = 4, axes = TRUE, graticule = TRUE) # rc_sf[1] - plotar a primeira coluna
```



# 3. Classes sf

## 3. Simple feature (sf)

```
# plot  
plot(rc_sf[2], pch = 20, cex = 4, axes = TRUE, graticule = TRUE) # rc_sf[2] - plotar a segunda coluna
```



# 3. Classes sf

## 3. Simple feature (sf)

```
# plot  
plot(rc_sf$geometry, pch = 20, cex = 4, axes = TRUE, graticule = TRUE) # rc_sf$geometry - plotar apenas a geometria
```

# 4. Importar dados vetoriais

## Fundação Brasileira Desenvolvimento Sustentável (FBDS)

Em 2015, a FBDS deu início ao *Projeto de Mapeamento em Alta Resolução dos Biomas Brasileiros*:

- Cobertura da terra
- Hidrografia (nascentes, rios e lagos)
- Áreas de Preservação Permanente (APP)

O mapeamento foi concluído para os municípios dos biomas *Mata Atlântica e Cerrado*

Site: <https://www.fbds.org.br/>

Repositório: <http://geo.fbds.org.br/>



# 4. Importar dados vetoriais

## Download

### Criar um diretório

```
# create directory
dir.create(here::here("03_dados", "vetor"))
```

### Download de **pontos de nascentes** para Rio Claro/SP

```
# increase time to download
options(timeout = 600)

# download points
for(i in c(".dbf", ".prj", ".shp", ".shx")){
  download.file(url = paste0("http://geo.fbds.org.br/SP/RIO_CLARO/HIDROGRAFIA/SP_3543907_NASCENTES", i),
               destfile = here::here("03_dados", "vetor", paste0("SP_3543907_NASCENTES", i)), mode = "wb")
}
```



# 4. Importar dados vetoriais

## Download

### Download de **linhas da hidrografia** para Rio Claro/SP

```
# download lines
for(i in c(".dbf", ".prj", ".shp", ".shx")){
  download.file(url = paste0("http://geo.fbds.org.br/SP/RIO_CLARO/HIDROGRAFIA/SP_3543907_RIOS_SIMPLES", i),
    destfile = here::here("03_dados", "vetor", paste0("SP_3543907_RIOS_SIMPLES", i)), mode = "wb")
}
```

### Download de **polígonos de cobertura da terra** para Rio Claro/SP

```
# download polygon
for(i in c(".dbf", ".prj", ".shp", ".shx")){
  download.file(url = paste0("http://geo.fbds.org.br/SP/RIO_CLARO/USO/SP_3543907_USO", i),
    destfile = here::here("03_dados", "vetor", paste0("SP_3543907_USO", i)), mode = "wb")
}
```

# 4. Importar dados vetoriais

## Dados preexistentes

```
# importar pontos
rc_nas <- sf::st_read(here::here("03_dados", "vetor", "SP_3543907_NASCENTES.shp"), quiet = TRUE)
rc_nas
```

```
## Simple feature collection with 1220 features and 5 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:  xmin: 217622.9 ymin: 7504132 xmax: 246367.4 ymax: 7537855
## Projected CRS: SIRGAS 2000 / UTM zone 23S
## First 10 features:
##      GEOCODIGO MUNICIPIO UF CD_UF      HIDRO      geometry
## 1      3543907 RIO CLARO SP      35 nascente POINT (217622.9 7528315)
## 2      3543907 RIO CLARO SP      35 nascente POINT (217836.5 7528103)
## 3      3543907 RIO CLARO SP      35 nascente POINT (217988.9 7528203)
## 4      3543907 RIO CLARO SP      35 nascente POINT (218288.9 7528237)
## 5      3543907 RIO CLARO SP      35 nascente POINT (218346.6 7530583)
## 6      3543907 RIO CLARO SP      35 nascente POINT (218393.1 7528031)
## 7      3543907 RIO CLARO SP      35 nascente POINT (218508.3 7528470)
## 8      3543907 RIO CLARO SP      35 nascente POINT (218535.4 7530642)
## 9      3543907 RIO CLARO SP      35 nascente POINT (218583.5 7528215)
```

## 4. Importar dados vetoriais

### Dados preexistentes

```
# plot  
plot(rc_nas$geometry, pch = 20, col = "blue", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados preexistentes

```
# importar linhas
rc_hid ← sf::st_read(here::here("03_dados", "vetor", "SP_3543907_RIOS_SIMPLES.shp"), quiet = TRUE)
rc_hid
```

```
## Simple feature collection with 1 feature and 6 fields
## Geometry type: MULTILINESTRING
## Dimension:      XY
## Bounding box:   xmin: 215155.3 ymin: 7504132 xmax: 246367.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          HIDRO COMP_KM          geometry
## 1   3543907 RIO CLARO SP     35 curso d'água (0 - 10m) 1142.98 MULTILINESTRING ((231815.7 ...
```

## 4. Importar dados vetoriais

### Dados preexistentes

```
# plot  
plot(rc_hid$geometry, col = "steelblue", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados preexistentes

```
# importar poligonos
rc_cob <- sf::st_read(here::here("03_dados", "vetor", "SP_3543907_USO.shp"), quiet = TRUE)
rc_cob
```

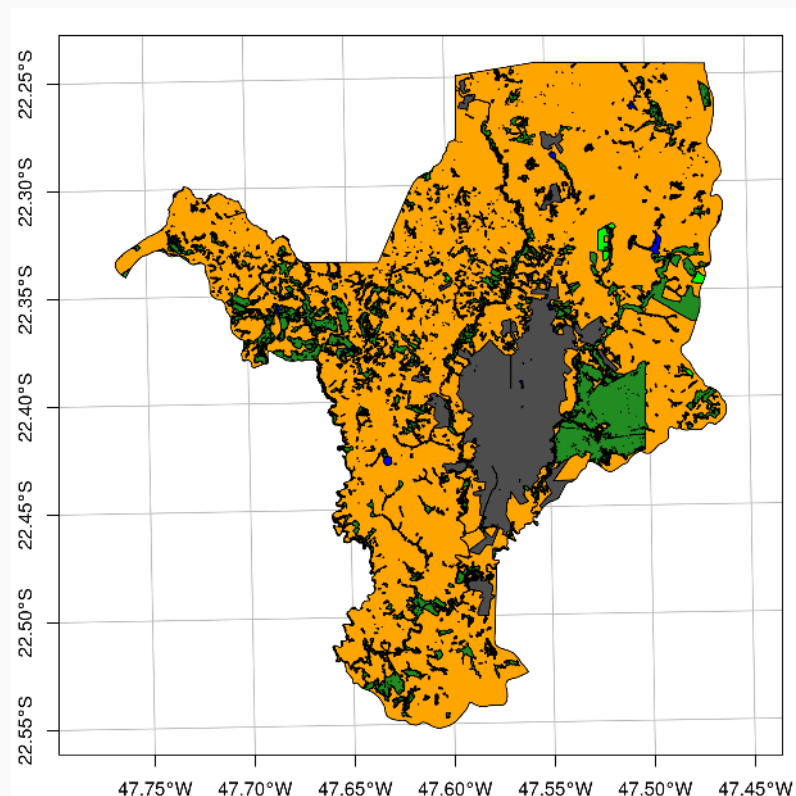
```
## Simple feature collection with 5 features and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:   xmin: 215151.7 ymin: 7503723 xmax: 246582.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

##	GEOCODIGO	MUNICIPIO	UF	CD_UF	CLASSE_USO	AREA_HA	geometry
## 1	3543907	RIO CLARO	SP	35	água	357.027	MULTIPOLYGON (((235487.6 75 ...
## 2	3543907	RIO CLARO	SP	35	área antropizada	37297.800	MULTIPOLYGON (((232275 7504 ...
## 3	3543907	RIO CLARO	SP	35	área edificada	5078.330	MULTIPOLYGON (((233123.6 75 ...
## 4	3543907	RIO CLARO	SP	35	formação florestal	7017.990	MULTIPOLYGON (((232355 7504 ...
## 5	3543907	RIO CLARO	SP	35	silvicultura	138.173	MULTIPOLYGON (((243052.1 75 ...

# 4. Importar dados vetoriais

## Dados preexistentes

```
# plot  
plot(rc_cob[5], col = c("blue", "orange", "gray30", "forestgreen", "green"), main = NA, axes = TRUE, graticule =
```



# 4. Importar dados vetoriais

## Dados de GPS (.gpx)

```
# importar dados de gps
gps_gpx ← sf::st_read(here::here("03_dados", "vetor", "waypoints.gpx"), layer = "waypoints", quiet = TRUE)
gps_gpx
```

```
## Simple feature collection with 11 features and 23 fields
```

```
## Geometry type: POINT
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -47.54245 ymin: -22.39524 xmax: -47.53969 ymax: -22.39209
```

```
## Geodetic CRS: WGS 84
```

```
## First 10 features:
```

##	ele	time	magvar	geoidheight	name	cmt	desc	src	link1_href	link1_text	link1_type	link2_href
## 1	603.0454	2020-08-20 10:13:35	NA	NA	102	33	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 2	595.2947	2020-08-20 10:22:52	NA	NA	103	71	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 3	581.9885	2020-08-20 10:32:30	NA	NA	104	11	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 4	587.4568	2020-08-20 10:42:05	NA	NA	105	72	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 5	581.2265	2020-08-20 10:49:32	NA	NA	106	68	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 6	593.6844	2020-08-20 11:06:53	NA	NA	107	121	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 7	591.4097	2020-08-20 11:16:36	NA	NA	108	83	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 8	594.1143	2020-08-20 11:27:59	NA	NA	109	162	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>
## 9	594.4582	2020-08-20 11:43:04	NA	NA	110	103	<NA>	<NA>	<NA>	<NA>	<NA>	<NA>



## 4. Importar dados vetoriais

### Dados de GPS (.gpx)

```
# plot  
plot(gps_gpx$geometry, cex = 4, pch = 20, col = "red", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de GPS (.kml)

```
# importar dados de gps
gps_kml ← sf::st_read(here::here("03_dados", "vetor", "waypoints.kml"), quiet = TRUE)
gps_kml
```

```
## Simple feature collection with 11 features and 16 fields
```

```
## Geometry type: POINT
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -47.54245 ymin: -22.39524 xmax: -47.53969 ymax: -22.39209
```

```
## Geodetic CRS: WGS 84
```

```
## First 10 features:
```

##	Name	description	timestamp	begin	end	altitudeMode	tessellate	extrude	visibility	drawOrder	icon	elevation	comme
## 1	102	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	603.0454	
## 2	103	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	595.2947	
## 3	104	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	581.9885	
## 4	105	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	587.4568	
## 5	106	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	581.2265	
## 6	107	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	593.6844	1
## 7	108	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	591.4097	
## 8	109	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	594.1143	1
## 9	110	<NA>	<NA>	<NA>	<NA>	<NA>	-1	0	-1	NA	<NA>	594.4582	1

## 4. Importar dados vetoriais

### Dados de GPS (.kml)

```
# plot  
plot(gps_kml$geometry, cex = 4, pch = 20, col = "blue", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de uma tabela de coordenadas

```
# importar tabela
si <- readr::read_csv(
  here::here("03_dados", "tabelas", "ATLANTIC_AMPHIBIANS_sites.csv"))
si
```

```
## # A tibble: 1,163 × 25
##   id      reference_number species_number record sampled_habitat active_methods passive_methods complementary_me...
##   <chr> <chr> <dbl> <dbl> <chr> <chr> <chr> <chr>
## 1 amp1001 001 19 ab fo,ll as pt
## 2 amp1002 002 16 co fo,la,ll as pt
## 3 amp1003 002 14 co fo,la,ll as pt
## 4 amp1004 002 13 co fo,la,ll as pt
## 5 amp1005 003 30 co fo,ll,br as NA
## 6 amp1006 004 42 co tp,pp,la,ll,is NA
## 7 amp1007 005 23 co sp as NA
## 8 amp1008 005 19 co sp,la,sw as,sb,tr NA
## 9 amp1009 005 13 ab fo NA pt
## 10 amp1010 006 1 ab fo NA pt
## ... with 1,153 more rows, and 14 more variables: month_finish <dbl>, year_finish <dbl>, effort_months <dbl>,
##   state_abbreviation <chr>, municipality <chr>, site <chr>, latitude <dbl>, longitude <dbl>, coordinate_pre
```

# 4. Importar dados vetoriais

## Dados de uma tabela de coordenadas (.csv | .xlsx)

```
# converter para sf
si_ve <- si %>%
  sf::st_as_sf(coords = c("longitude", "latitude"), crs = 4326)
si_ve
```

```
## Simple feature collection with 1163 features and 23 fields
```

```
## Geometry type: POINT
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -56.74194 ymin: -33.51083 xmax: -34.79667 ymax: -3.51525
```

```
## Geodetic CRS: WGS 84
```

```
## # A tibble: 1,163 × 24
```

```
##   id      reference_number species_number record sampled_habitat active_methods passive_methods complementary_me...
```

```
##   <dbl> <dbl> <dbl> <dbl> <chr> <dbl> <dbl> <dbl> <chr>
```

```
## 1 1001 1001 19 ab fo,ll as pt
```

```
## 2 1002 1002 16 co fo,la,ll as pt
```

```
## 3 1003 1002 14 co fo,la,ll as pt
```

```
## 4 1004 1002 13 co fo,la,ll as pt
```

```
## 5 1005 1003 30 co fo,ll,br as NA
```

```
## 6 1006 1004 42 co tp,pp,la,ll,is NA
```

```
## 7 1007 1005 23 co sp as NA
```

## 4. Importar dados vetoriais

Dados de uma tabela de coordenadas (.csv | .xlsx)

```
# plot  
plot(si_ve$geometry, pch = 20, main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
install.packages("geobr")  
library(geobr)
```



# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
# brasil 2020
br_2020 <- geobr::read_country(year = 2020, showProgress = FALSE)
br_2020
```

## Simple feature collection with 27 features and 5 fields

## Geometry type: MULTIPOLYGON

## Dimension: XY

## Bounding box: xmin: -73.99045 ymin: -33.75118 xmax: -28.84784 ymax: 5.271841

## Geodetic CRS: SIRGAS 2000

## First 10 features:

##	code_state	abbrev_state	name_state	code_region	name_region	geom
## 1	11	RO	Rondônia	1	Norte	MULTIPOLYGON (((-65.3815 -1 ...
## 2	12	AC	Acre	1	Norte	MULTIPOLYGON (((-71.07772 - ...
## 3	13	AM	Amazônas	1	Norte	MULTIPOLYGON (((-69.83766 - ...
## 4	14	RR	Roraima	1	Norte	MULTIPOLYGON (((-63.96008 2 ...
## 5	15	PA	Pará	1	Norte	MULTIPOLYGON (((-51.43248 - ...
## 6	16	AP	Amapá	1	Norte	MULTIPOLYGON (((-53.27918 2 ...
## 7	17	TO	Tocantins	1	Norte	MULTIPOLYGON (((-48.23291 - ...
## 8	21	MA	Maranhão	2	Nordeste	MULTIPOLYGON (((-46.16128 - ...
## 9	22	PI	Piauí	2	Nordeste	MULTIPOLYGON (((-42.91509 - ...



## 4. Importar dados vetoriais

Dados de pacotes: *geobr*

```
# plot  
plot(br_2020$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
# brasil 1872
br_1872 ← geobr::read_country(year = 1872, showProgress = FALSE)
br_1872

## Simple feature collection with 1 feature and 0 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:   xmin: -73.80132 ymin: -33.74912 xmax: -34.79313 ymax: 5.271891
## Geodetic CRS:   SIRGAS 2000
##
##              geom
## 1 MULTIPOLYGON (((-48.40975 - ...
```

## 4. Importar dados vetoriais

Dados de pacotes: *geobr*

```
# plot  
plot(br_1872$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
# sao paulo municipios
sp_mun_2020 <- geobr::read_municipality(code_muni = "SP", year = 2020, showProgress = FALSE)
sp_mun_2020
```

## Simple feature collection with 645 features and 7 fields

## Geometry type: MULTIPOLYGON

## Dimension: XY

## Bounding box: xmin: -53.10986 ymin: -25.35794 xmax: -44.16137 ymax: -19.77966

## Geodetic CRS: SIRGAS 2000

## First 10 features:

##	code_muni	name_muni	code_state	abbrev_state	name_state	code_region	name_region
## 1	3500105	Adamantina	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-51
## 2	3500204	Adolfo	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-49
## 3	3500303	Aguaí	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-46
## 4	3500402	Águas Da Prata	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-46
## 5	3500501	Águas De Lindóia	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-46
## 6	3500550	Águas De Santa Bárbara	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-49
## 7	3500600	Águas De São Pedro	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-47
## 8	3500709	Agudos	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-48
## 9	3500758	Alambari	35	SP	São Paulo	3	Sudeste MULTIPOLYGON (((-47

## 4. Importar dados vetoriais

Dados de pacotes: *geobr*

```
# plot  
plot(sp_mun_2020$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
# rio claro
```

```
rc_2020 ← geobr::read_municipality(code_muni = 3543907, year = 2020, showProgress = FALSE)
```

```
rc_2020
```

```
## Simple feature collection with 1 feature and 7 fields
```

```
## Geometry type: MULTIPOLYGON
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -47.76521 ymin: -22.55203 xmax: -47.46188 ymax: -22.24368
```

```
## Geodetic CRS: SIRGAS 2000
```

```
##      code_muni name_muni code_state abbrev_state name_state code_region name_region          geom
```

```
## 493    3543907 Rio Claro          35          SP  São Paulo          3      Sudeste MULTIPOLYGON (((-47.46875 - ...
```

## 4. Importar dados vetoriais

Dados de pacotes: *geobr*

```
# plot  
plot(rc_2020$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
```

# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
# biomas
bi_2019 ← geobr::read_biomes(year = 2019, showProgress = FALSE)
bi_2019
```

```
## Simple feature collection with 7 features and 3 fields
```

```
## Geometry type: MULTIPOLYGON
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -73.98304 ymin: -34.95942 xmax: -28.84785 ymax: 7.053767
```

```
## Geodetic CRS: SIRGAS 2000
```

```
##      name_biome code_biome year      geom
## 1      Amazônia      1 2019 MULTIPOLYGON (((-44.08515 - ...
## 2      Caatinga      2 2019 MULTIPOLYGON (((-41.7408 -2 ...
## 3      Cerrado      3 2019 MULTIPOLYGON (((-43.39009 - ...
## 4  Mata Atlântica      4 2019 MULTIPOLYGON (((-48.70814 - ...
## 5          Pampa      5 2019 MULTIPOLYGON (((-52.82472 - ...
## 6      Pantanal      6 2019 MULTIPOLYGON (((-57.75946 - ...
## 7 Sistema Costeiro    NA 2019 MULTIPOLYGON (((-44.64799 - ...
```



## 4. Importar dados vetoriais

Dados de pacotes: *geobr*

```
# plot  
plot(bi_2019$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
```

## 4. Importar dados vetoriais

Dados de pacotes: *geobr*

Function	Geographies available	Years available	Source
<code>read_country()</code>	Country	1872, 1900, 1911, 1920, 1933, 1940, 1950, 1960, 1970, 1980, 1991, 2000, 2001, 2010, 2013, 2014, 2015, 2016, 2017, 2018, 2019	IBGE
<code>read_state()</code>	States	1872, 1900, 1911, 1920, 1933, 1940, 1950, 1960, 1970, 1980, 1991, 2000, 2001, 2010, 2013, 2014, 2015, 2016, 2017, 2018, 2019	IBGE
<code>read_municipality()</code>	Municipality	1872, 1900, 1911, 1920, 1933, 1940, 1950, 1960, 1970, 1980, 1991, 2000, 2001, 2005, 2007, 2010, 2013, 2014, 2015, 2016, 2017, 2018, 2019	IBGE
<code>read_biomes()</code>	Biomes	2004, 2019	IBGE

# 4. Importar dados vetoriais

## Dados de pacotes: *geobr*

```
# list all datasets available in the geobr package
geobr::list_geobr()
```

```
## # A tibble: 27 × 4
##   `function`      geography      years
##   <chr>          <chr>      <chr>
## 1 `read_country`   Country    1872, 1900, 1911, 1920, 1933, 1
## 2 `read_region`   Region     2000, 2001, 2010, 2013, 2014, 2
## 3 `read_state`   States     1872, 1900, 1911, 1920, 1933, 1
## 4 `read_meso_region` Meso region 2000, 2001, 2010, 2013, 2014, 2
## 5 `read_micro_region` Micro region 2000, 2001, 2010, 2013, 2014, 2
## 6 `read_intermediate_region` Intermediate region 2017, 2019, 2020
## 7 `read_immediate_region` Immediate region 2017, 2019, 2020
## 8 `read_municipality` Municipality 1872, 1900, 1911, 1920, 1933, 1
## 9 `read_municipal_seat` Municipality seats (sedes municipais) 1872, 1900, 1911, 1920, 1933, 1
## 10 `read_weighting_area` Census weighting area (área de ponderação) 2010
## # ... with 17 more rows
```

## 4. Importar dados vetoriais

Dados de pacotes: *rnaturalearth*

```
# package  
install.packages("rnaturalearth")  
library(rnaturalearth)
```



[naturalearthdata](#), [rnaturalearth](#)

# 4. Importar dados vetoriais

## Dados de pacotes: *rnaturalearth*

```
# south america
sa <- rnaturalearth::ne_countries(scale = "small", continent = "South America", returnclass = "sf")
sa
```

```
## Simple feature collection with 13 features and 63 fields
```

```
## Geometry type: MULTIPOLYGON
```

```
## Dimension: XY
```

```
## Bounding box: xmin: -81.41094 ymin: -55.61183 xmax: -34.72999 ymax: 12.4373
```

```
## CRS: +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
```

```
## First 10 features:
```

##	scalerank	featurecla	labelrank	sovereignty	sov_a3	adm0_dif	level	type	admin ad
## 4	1	Admin-0 country	2	Argentina	ARG	0	2	Sovereign country	Argentina
## 21	1	Admin-0 country	3	Bolivia	BOL	0	2	Sovereign country	Bolivia
## 22	1	Admin-0 country	2	Brazil	BRA	0	2	Sovereign country	Brazil
## 29	1	Admin-0 country	2	Chile	CHL	0	2	Sovereign country	Chile
## 35	1	Admin-0 country	2	Colombia	COL	0	2	Sovereign country	Colombia
## 46	1	Admin-0 country	3	Ecuador	ECU	0	2	Sovereign country	Ecuador
## 54	1	Admin-0 country	5	United Kingdom	GB1	1	2	Dependency	Falkland Islands
## 67	1	Admin-0 country	4	Guyana	GUY	0	2	Sovereign country	Guyana
## 124	1	Admin-0 country	2	Peru	PER	0	2	Sovereign country	Peru

## 4. Importar dados vetoriais

Dados de pacotes: *rnaturalearth*

```
# plot  
plot(sa$geometry, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
```

# 5. Descrição de objetos sf

## Informações geográficas e tabela de atributos

```
# rio claro  
rc_2020
```

```
## Simple feature collection with 1 feature and 7 fields  
## Geometry type: MULTIPOLYGON  
## Dimension:      XY  
## Bounding box:  xmin: -47.76521 ymin: -22.55203 xmax: -47.46188 ymax: -22.24368  
## Geodetic CRS:  SIRGAS 2000  
##      code_muni name_muni code_state abbrev_state name_state code_region name_region          geom  
## 493    3543907 Rio Claro          35           SP  São Paulo          3      Sudeste MULTIPOLYGON (((-47.46875 - ...
```

# 5. Descrição de objetos sf

## Informações

- *resumo do vetor*: indica o número de feições (linhas) e campos (colunas)
- *tipo da geometria*: umas das sete classes listadas anteriormente
- *dimensão*: número de dimensões, geralmente duas (XY)
- *bbox (bordas)*: coordenadas mínimas e máximas da longitude e latitude
- *informação do CRS*: `epsg` ou `proj4string` indicando o CRS
- *tibble*: tabela de atributos, com destaque para a coluna `geom` ou `geometry` que representa cada feição ou geometria

```
## 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...
```

Simple feature

Simple feature geometry list-column (sfc)

Simple feature geometry (sfg)



# 5. Descrição de objetos sf

## Informações geográficas

### Geometry type

```
# tipo de geometria  
sf::st_geometry_type(rc_2020)
```

```
## [1] MULTIPOLYGON
```

```
## 18 Levels: GEOMETRY POINT LINESTRING POLYGON MULTIPOINT MULTILINESTRING MULTIPOLYGON GEOMETRYCOLLECTION CIRCULARST
```

### Bounding box

```
# extensao  
sf::st_bbox(rc_2020)
```

```
##      xmin      ymin      xmax      ymax
```

```
## -47.76521 -22.55203 -47.46188 -22.24368
```

# 5. Descrição de objetos sf

## Informações geográficas

### Coordinate Reference System (CRS)

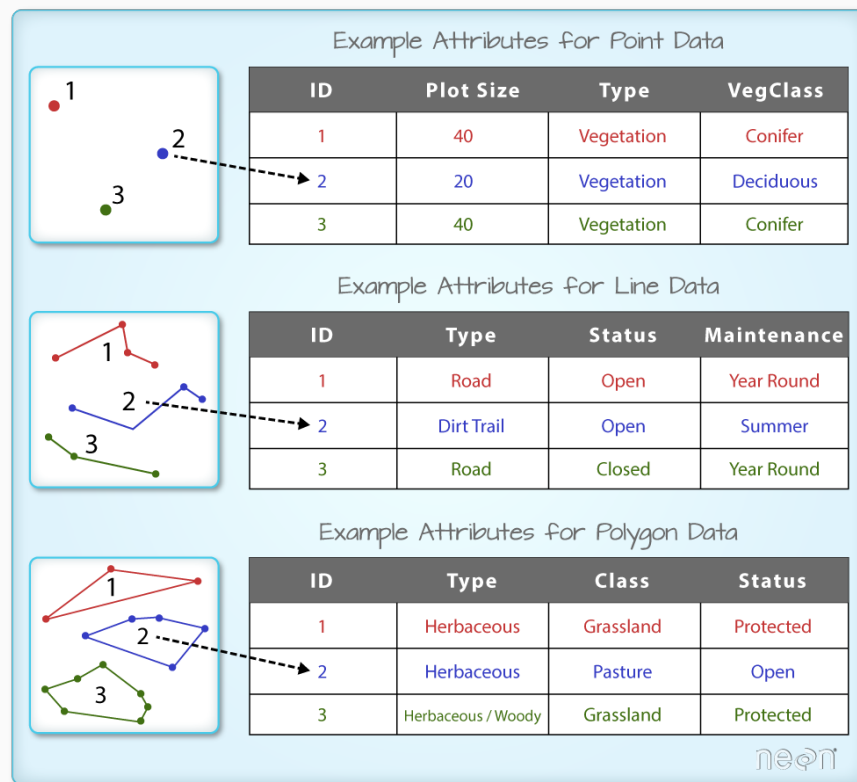
```
# crs
sf::st_crs(rc_2020)
```

```
## Coordinate Reference System:
##   User input: SIRGAS 2000
##   wkt:
##   GEOGCRS["SIRGAS 2000",
##     DATUM["Sistema de Referencia Geocentrico para las AmericaS 2000",
##       ELLIPSOID["GRS 1980",6378137,298.257222101,
##         LENGTHUNIT["metre",1]]],
##     PRIMEM["Greenwich",0,
##       ANGLEUNIT["degree",0.0174532925199433]],
##     CS[ellipsoidal,2],
##       AXIS["geodetic latitude (Lat)",north,
##         ORDER[1],
##         ANGLEUNIT["degree",0.0174532925199433]],
##       AXIS["geodetic longitude (Lon)",east,
##         ORDER[2],
```

# 5. Descrição de objetos sf

## Tabela de atributos

## Relação entre a geometria e suas características



# 5. Descrição de objetos sf

## Tabela de atributos

Acessar a tabela de atributos de um `sf`

```
# acessar a tabela de atributos  
rc_2020_tab ← sf::st_drop_geometry(rc_2020)  
rc_2020_tab
```

```
##      code_muni name_muni code_state abbrev_state name_state code_region name_region  
## 493    3543907 Rio Claro         35          SP   São Paulo          3      Sudeste
```

```
# classe  
class(rc_2020_tab)
```

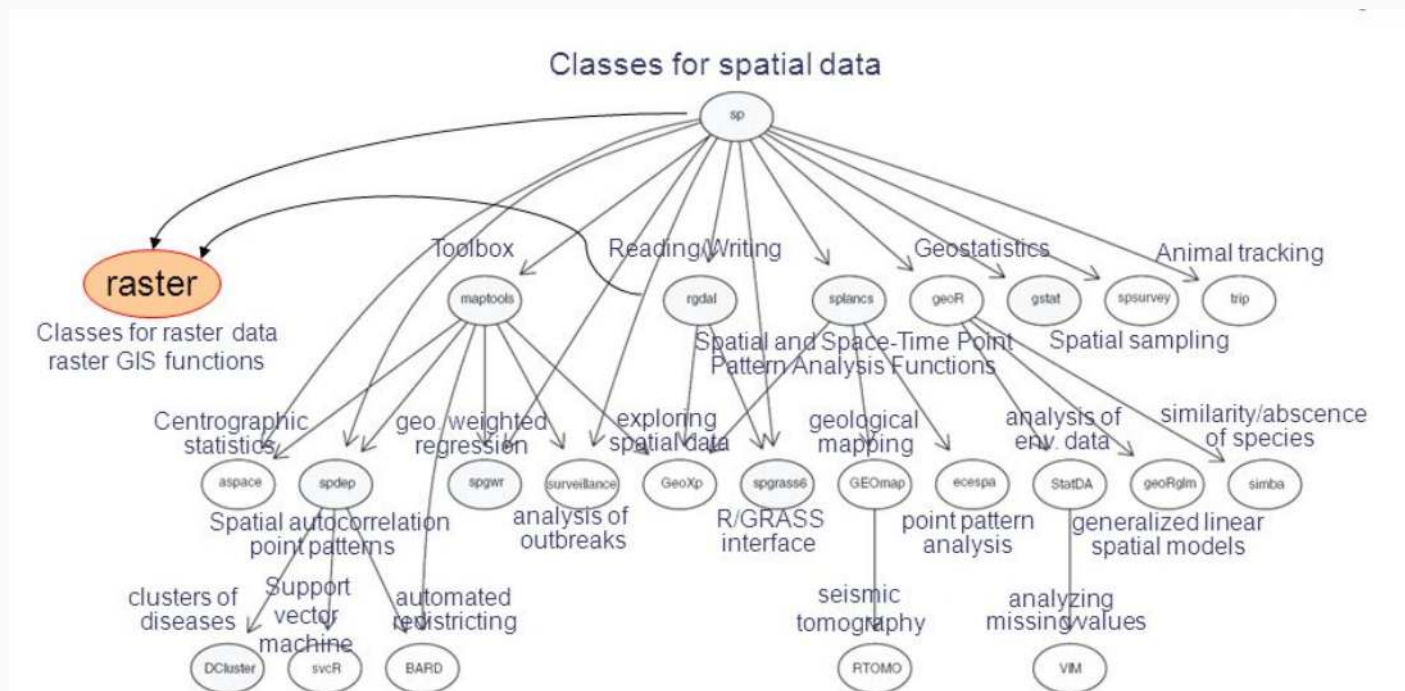
```
## [1] "data.frame"
```

# 6. Converter objetos para sf

## Tipos de objetos `sp`

- SpatialPoints
- SpatialLines
- SpatialPolygons
- SpatialPointsdata frame
- SpatialLinesdata frame
- SpatialPolygonsdata frame

R package	Class
maps [8]	map
sp [10]	SpatialPoints SpatialPointsDataFrame Line Lines SpatialLines SpatialLinesDataFrame Polygon Polygons SpatialPolygons SpatialPolygonsDataFrame



# 6. Converter objetos para sf

## Dados `sp`

```
# paises sp
co110_sp <- rnatualearth::countries110
co110_sp
```

```
## class      : SpatialPolygonsDataFrame
## features   : 177
## extent     : -180, 180, -90, 83.64513 (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs +ellps=WGS84 +towgs84=0,0,0
## variables  : 63
## names      : scalerank,      featurecla, labelrank,  sovereignt, sov_a3, adm0_dif, level,      type,
## min values :      1, Admin-0 country,      2, Afghanistan,  AFG,      0,      2,      Country, Afgh
## max values :      3, Admin-0 country,      7,      Zimbabwe,  ZWE,      1,      2, Sovereign country,  Z
```

# 6. Converter objetos para sf

## Converter `sp` para `sf`

```
# paises sf
co110_sf <- sf::st_as_sf(co110_sp)
co110_sf
```

```
## Simple feature collection with 177 features and 63 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
## First 10 features:
```

##	scalerank	featurecla	labelrank	sovereignty	sov_a3	adm0_dif	level	type
## 0	1	Admin-0 country	3	Afghanistan	AFG	0	2	Sovereign country
## 1	1	Admin-0 country	3	Angola	AGO	0	2	Sovereign country
## 2	1	Admin-0 country	6	Albania	ALB	0	2	Sovereign country
## 3	1	Admin-0 country	4	United Arab Emirates	ARE	0	2	Sovereign country
## 4	1	Admin-0 country	2	Argentina	ARG	0	2	Sovereign country
## 5	1	Admin-0 country	6	Armenia	ARM	0	2	Sovereign country
## 6	1	Admin-0 country	4	Antarctica	ATA	0	2	Indeterminate
## 7	3	Admin-0 country	6	France	FR1	1	2	Dependency French Southern
## 8	1	Admin-0 country	2	Australia	AU1	1	2	Country

## 6. Converter objetos para sf

### Converter `sf` para `sp`

```
# plot  
plot(col10_sf$geometry, col = "gray", main = NA, graticule = TRUE)
```



# 6. Converter objetos para sf

## Converter `sf` para `sp`

```
# paises sp
co110_sp <- sf::as_Spatial(co110_sf)
co110_sp
```

```
## class      : SpatialPolygonsDataFrame
## features   : 177
## extent     : -180, 180, -90, 83.64513 (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +ellps=WGS84 +towgs84=0,0,0,0,0,0,0 +no_defs
## variables  : 63
## names      : scalerank,      featurecla, labelrank,  sovereignty, sov_a3, adm0_dif, level,      type,
## min values :      1, Admin-0 country,      2, Afghanistan,  AFG,      0,      2,      Country, Afgh
## max values :      3, Admin-0 country,      7,      Zimbabwe,  ZWE,      1,      2, Sovereign country,  Z
```

# 7. Converter CRS de objetos sf

## Os principais CRSs

CRS	Tipo de CRS	Descrição	epsg.io	spatialreference.org
WGS84	Geográfico	CRS geográfico mais comum para o mundo	[EPSG:4326] ( <a href="http://epsg.io/4326">http://epsg.io/4326</a> )	[EPSG:4326] ( <a href="https://spatialreference.org/ref/epsg/4326/">https://spatialreference.org/ref/epsg/4326/</a> )
SIRGAS 2000	Geográfico	CRS geográfico oficial para o Brasil	[EPSG:4674] ( <a href="http://epsg.io/4674">http://epsg.io/4674</a> )	[EPSG:4674] ( <a href="https://spatialreference.org/ref/epsg/4674/">https://spatialreference.org/ref/epsg/4674/</a> )
Projeção de Mollweide	Projetado	CRS projetado que preserva as relações de área	[ESRI:54009] ( <a href="https://epsg.io/54009">https://epsg.io/54009</a> )	[SR-ORG:7099] ( <a href="https://spatialreference.org/ref/sr-org/7099/">https://spatialreference.org/ref/sr-org/7099/</a> )
Projeção de Winkel Tripel	Projetado	CRS projetado que preserva a área e com meridianos elípticos	[EPSG:54012] ( <a href="https://epsg.io/54012">https://epsg.io/54012</a> )	[ESRI:54012] ( <a href="https://spatialreference.org/ref/esri/54012/">https://spatialreference.org/ref/esri/54012/</a> )

# 7. Converter CRS de objetos sf

## Converter CRS local

SIRGAS2000/GCS -> SIRGAS2000/UTM23S

```
# converter sistema de coordenadas  
rc_2020_sirgas2000_utm23s ← sf::st_transform(rc_2020, crs = 31983)
```

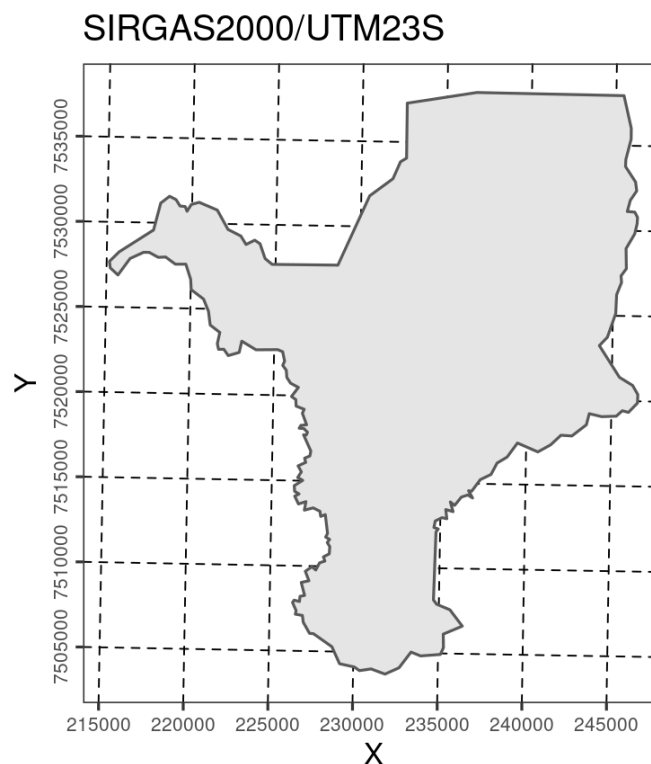
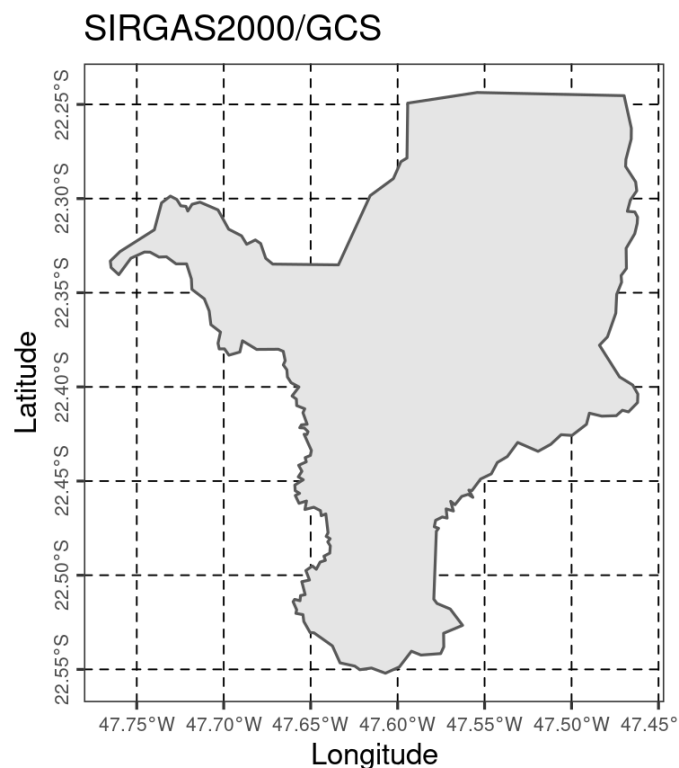


[EPSG: 31983](#) [SIRGAS2000](#)

# 7. Converter CRS de objetos sf

## Converter CRS local

SIRGAS2000/GCS -> SIRGAS2000/UTM23S

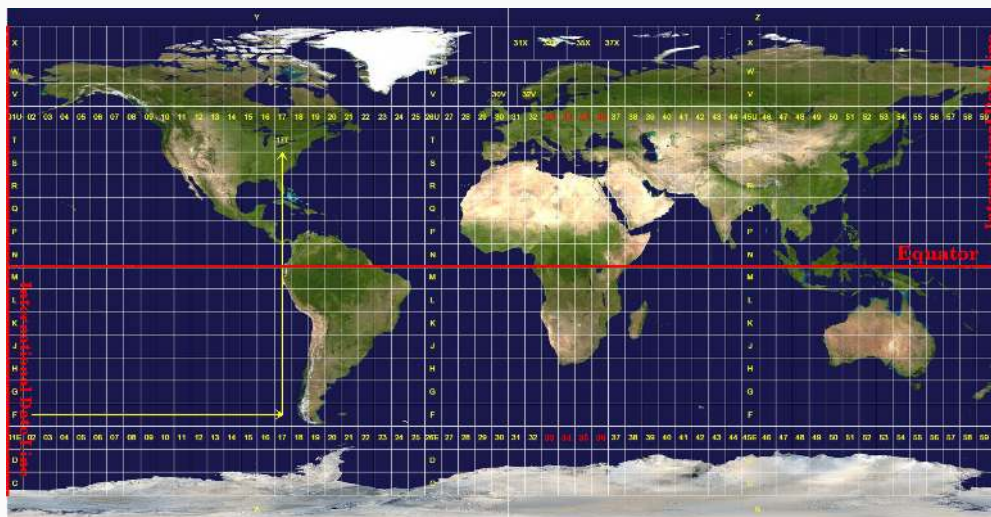


# 7. Converter CRS de objetos sf

## Converter CRS local

SIRGAS2000/GCS -> WGS84/UTM23S

```
# converter sistema de coordenadas e datum  
rc_2020_wgs84_utm23s ← sf::st_transform(rc_2020, crs = 32723)
```



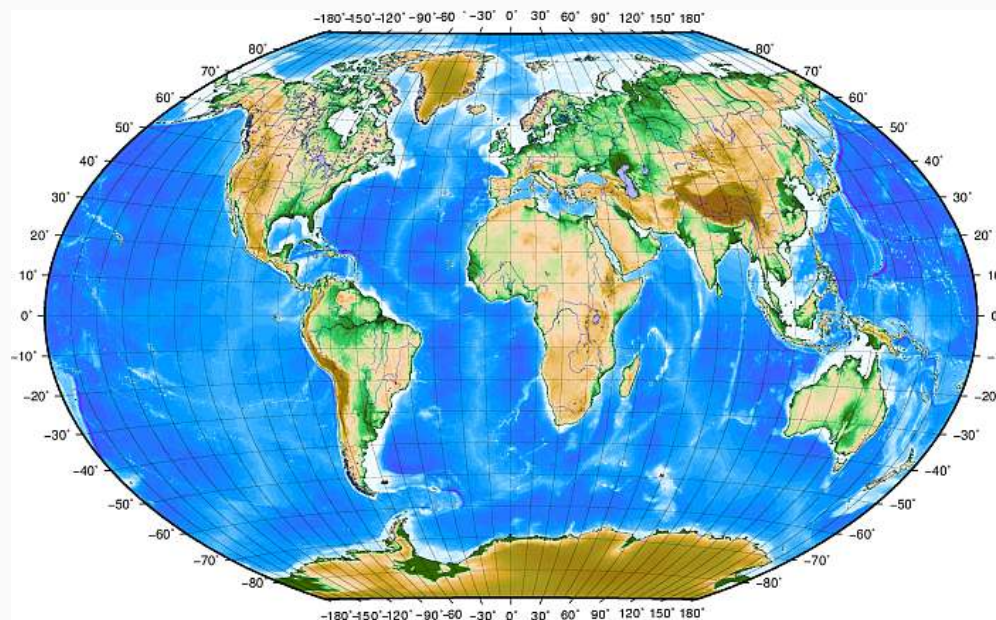
[EPSG: 32723](#) [ESRI \(2021\)](#)

# 7. Converter CRS de objetos sf

## Converter CRS local

SIRGAS2000/GCS -> WGS84/GCS

```
# converter datum  
rc_2020_wgs84_gcs ← sf::st_transform(rc_2020, crs = 4326)
```



[EPSG: 4326](https://epsg.org/epsg/4326)

# 7. Converter CRS de objetos sf

## Converter CRS global

```
# paises - WGS84/GCS
co110_sf
```

```
## Simple feature collection with 177 features and 63 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
```

```
## First 10 features:
```

##	scalerank	featurecla	labelrank	sovereignty	sov_a3	adm0_dif	level	type
## 0	1	Admin-0 country	3	Afghanistan	AFG	0	2	Sovereign country
## 1	1	Admin-0 country	3	Angola	AGO	0	2	Sovereign country
## 2	1	Admin-0 country	6	Albania	ALB	0	2	Sovereign country
## 3	1	Admin-0 country	4	United Arab Emirates	ARE	0	2	Sovereign country
## 4	1	Admin-0 country	2	Argentina	ARG	0	2	Sovereign country
## 5	1	Admin-0 country	6	Armenia	ARM	0	2	Sovereign country
## 6	1	Admin-0 country	4	Antarctica	ATA	0	2	Indeterminate
## 7	3	Admin-0 country	6	France	FR1	1	2	Dependency French Southern
## 8	1	Admin-0 country	2	Australia	AU1	1	2	Country
## 9	1	Admin-0 country	4	Austria	AUT	0	2	Sovereign country

# 7. Converter CRS de objetos sf

## Converter CRS global

```
# plot  
plot(col10_sf$geometry, col = "gray", graticule = TRUE)
```

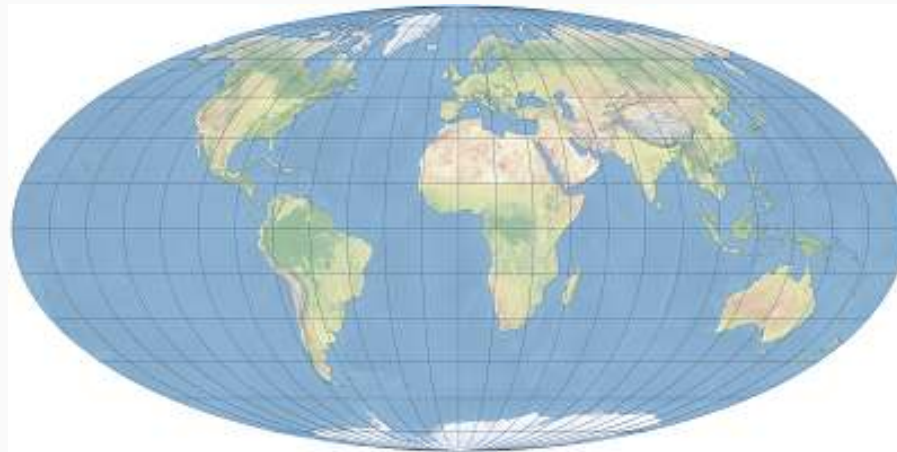


# 7. Converter CRS de objetos sf

## Converter CRS global

Projeção de Mollweide: preserva as relações de área

```
# projecao mollweide  
co110_sf_moll <- sf::st_transform(co110_sf, crs = "+proj=moll")
```



[EPSG: 54009](#) [ESRI \(2021\)](#).

# 7. Converter CRS de objetos sf

## Converter CRS global

### Projeção de Mollweide

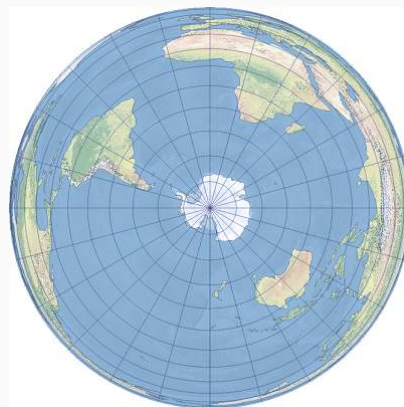
```
# plot  
plot(co110_sf_moll$geometry, col = "gray", graticule = TRUE)
```

# 7. Converter CRS de objetos sf

## Converter CRS global

Projeção azimutal de Lambert: preserva os tamanhos relativos e senso de direção a partir do centro

```
# projecao lambert  
co110_sf_laea ← sf::st_transform(co110_sf, crs = "+proj=laea +x_0=0 +y_0=0 +lon_0=0 +lat_0=0")
```



[ESRI\(2021\).](#)

# 7. Converter CRS de objetos sf

## Converter CRS global

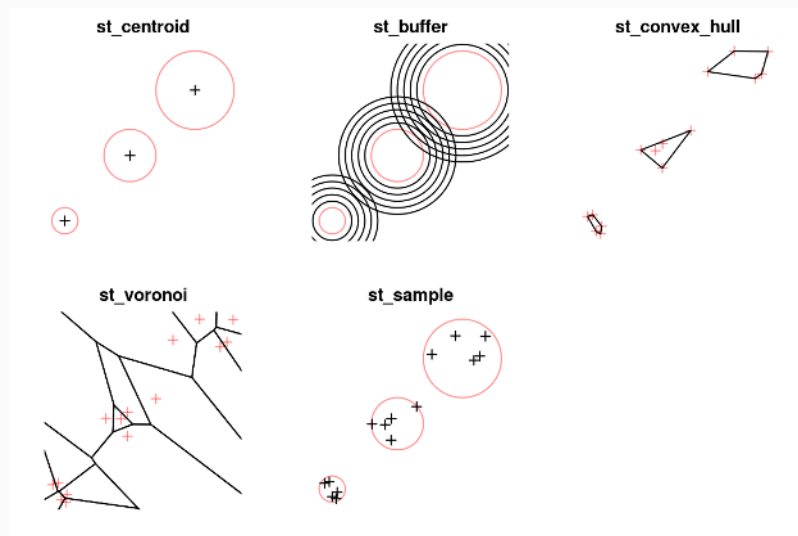
### Projeção azimutal de Lambert

```
# plot  
plot(co110_sf_laea$geometry, col = "gray", graticule = TRUE)
```

## 8. Operações com objetos sf

As operações podem ser separadas em três tipos

1. **Operações de atributos** (informações não espaciais, como tabela de atributos)
2. **Operações espaciais** (informações espaciais, como localização e formato)
3. **Operações geométricas** (informações geométricas, como mudanças geométricas unárias e binárias)



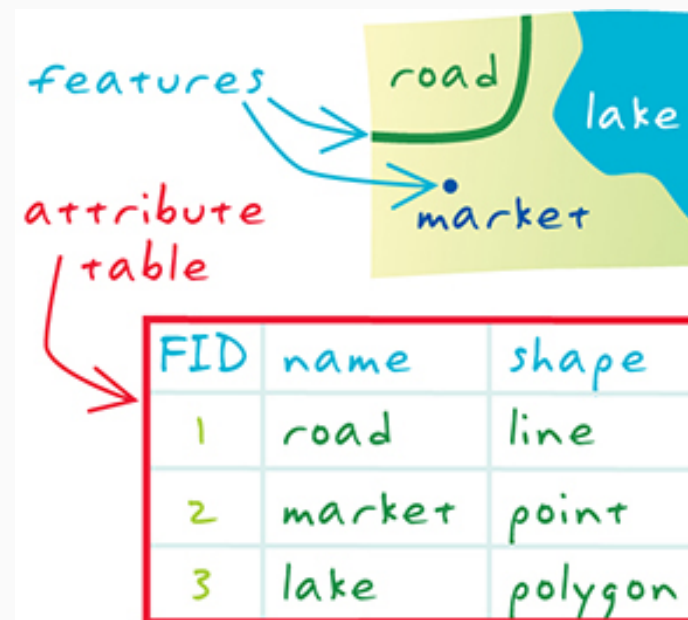
# 8. Operações com objetos sf

## 1. Operações de atributos

Modificação de objetos espaciais baseado em **informações não espaciais** associadas a objetos sf

### Operações

1. Filtro
2. Junção
3. Agregação
4. Manipulação da tabela de atributos



# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.1 Filtro

```
# filtro
rc_cob_floresta ← rc_cob %>%
  dplyr::filter(CLASSE_USO = "formação florestal")
rc_cob_floresta
```

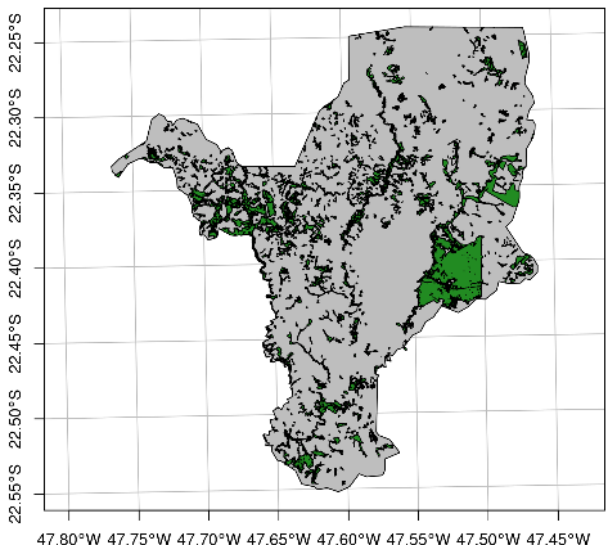
```
## Simple feature collection with 1 feature and 6 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:   xmin: 215442.4 ymin: 7504235 xmax: 246282.3 ymax: 7537969
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          CLASSE_USO AREA_HA              geometry
## 1   3543907 RIO CLARO SP    35 formação florestal 7017.99 MULTIPOLYGON (((232355 7504 ...
```

# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.1 Filtro

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_cob_floresta$geometry, col = "forestgreen", add = TRUE)
```





# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.2 Junção

```
# dados
da_classes <- tibble::tibble(CLASSE_USO = rc_cob$CLASSE_USO,
                             classe = c("agua", "antropico", "edificado", "floresta", "silvicultura"))
da_classes
```

```
## # A tibble: 5 × 2
##   CLASSE_USO      classe
##   <chr>         <chr>
## 1 água          agua
## 2 área antropizada antropico
## 3 área edificada edificado
## 4 formação florestal floresta
## 5 silvicultura  silvicultura
```

# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.2 Junção

```
# juncao
rc_cob_classes <- dplyr::left_join(rc_cob, da_classes, by = "CLASSE_USO") %>%
  sf::st_drop_geometry()
rc_cob_classes
```

##	GEOCODIGO	MUNICIPIO	UF	CD_UF	CLASSE_USO	AREA_HA	classe
## 1	3543907	RIO CLARO	SP	35	água	357.027	agua
## 2	3543907	RIO CLARO	SP	35	área antropizada	37297.800	antropico
## 3	3543907	RIO CLARO	SP	35	área edificada	5078.330	edificado
## 4	3543907	RIO CLARO	SP	35	formação florestal	7017.990	floresta
## 5	3543907	RIO CLARO	SP	35	silvicultura	138.173	silvicultura

# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.3 Agregação

```
# agregar
rc_nas_n <- rc_nas %>%
  dplyr::group_by(MUNICIPIO, HIDRO) %>%
  dplyr::summarise(n = n())
rc_nas_n
```

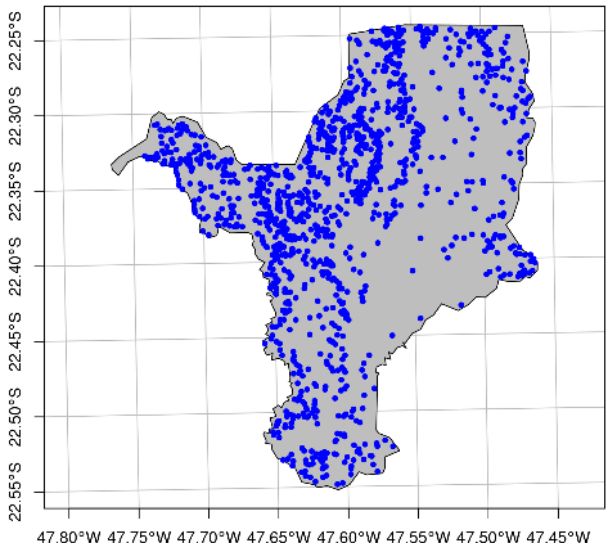
```
## Simple feature collection with 1 feature and 3 fields
## Geometry type: MULTIPOINT
## Dimension:      XY
## Bounding box:   xmin: 217622.9 ymin: 7504132 xmax: 246367.4 ymax: 7537855
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##  # A tibble: 1 × 4
##   MUNICIPIO HIDRO      n geomet
##   <chr>    <chr> <int> <sf>
## 1 RIO CLARO nascente 220  MULTIPOINT ((217622.9 7528315), (217836.5 7528103), (217988.9 7528203), (21828
```

# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.3 Agregação

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_nas_n$geometry, pch = 20, col = "blue", add = TRUE)
```



# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.4 Manipulação da tabela de atributos

```
# criar coluna
rc_cob_cob_col_area <- rc_cob %>%
  dplyr::mutate(classe_area = paste0(CLASSE_USO, " (", AREA_HA, " ha)")) %>%
  sf::st_drop_geometry()
rc_cob_cob_col_area
```

##	GEOCODIGO	MUNICIPIO	UF	CD_UF	CLASSE_USO	AREA_HA	classe_area
## 1	3543907	RIO CLARO	SP	35	água	357.027	água (357.027 ha)
## 2	3543907	RIO CLARO	SP	35	área antropizada	37297.800	área antropizada (37297.8 ha)
## 3	3543907	RIO CLARO	SP	35	área edificada	5078.330	área edificada (5078.33 ha)
## 4	3543907	RIO CLARO	SP	35	formação florestal	7017.990	formação florestal (7017.99 ha)
## 5	3543907	RIO CLARO	SP	35	silvicultura	138.173	silvicultura (138.173 ha)

# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.4 Manipulação da tabela de atributos

```
# comprimento das linhas
rc_hid_comp <- rc_hid %>%
  dplyr::mutate(comprimento = sf::st_length(.))
rc_hid_comp
```

```
## Simple feature collection with 1 feature and 7 fields
## Geometry type: MULTILINESTRING
## Dimension:      XY
## Bounding box:   xmin: 215155.3 ymin: 7504132 xmax: 246367.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          HIDRO COMP_KM      geometry comprimento
## 1   3543907 RIO CLARO SP    35 curso d'água (0 - 10m) 1142.98 MULTILINESTRING ((231815.7 ... 1142981 [m]
```

# 8. Operações com objetos sf

## 1. Operações de atributos

### 1.4 Manipulação da tabela de atributos

```
# area das poligonos
rc_cob_area <- rc_cob %>%
  dplyr::mutate(area_m2 = sf::st_area(.))
rc_cob_area
```

```
## Simple feature collection with 5 features and 7 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:   xmin: 215151.7 ymin: 7503723 xmax: 246582.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          CLASSE_USO  AREA_HA      geometry      area_m2
## 1   3543907 RIO CLARO SP    35             água    357.027 MULTIPOLYGON (((235487.6 75 ... 3570267 [m^2]
## 2   3543907 RIO CLARO SP    35   área antropizada 37297.800 MULTIPOLYGON (((232275 7504 ... 372978415 [m^2]
## 3   3543907 RIO CLARO SP    35   área edificada  5078.330 MULTIPOLYGON (((233123.6 75 ... 50783283 [m^2]
## 4   3543907 RIO CLARO SP    35 formação florestal 7017.990 MULTIPOLYGON (((232355 7504 ... 70179895 [m^2]
## 5   3543907 RIO CLARO SP    35      silvicultura  138.173 MULTIPOLYGON (((243052.1 75 ... 1381726 [m^2]
```

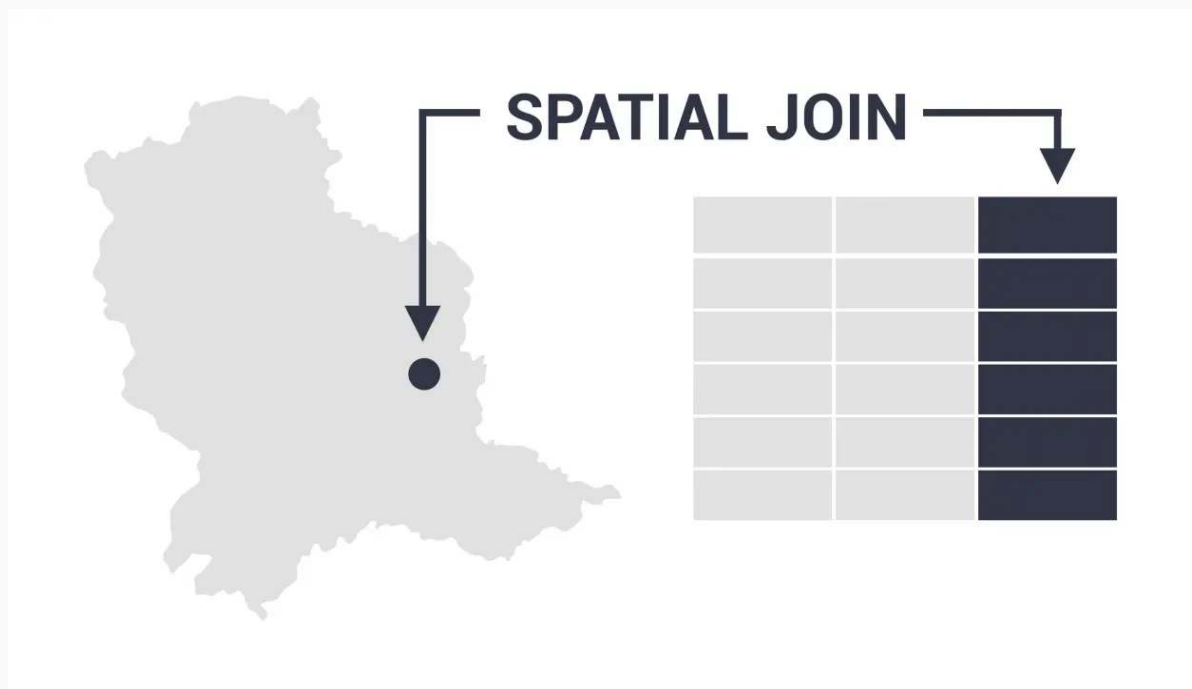
# 8. Operações com objetos sf

## 2 Operações espaciais

Modificação de objetos espaciais baseado na sua **localização e formato**

### Operações

1. Filtro espacial
2. Junção espacial
3. Agregação espacial
4. Distância espacial





# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial

```
# filtro espacial  
sf::st_intersects(x = rc_nas, y = rc_cob_floresta)
```

```
## Sparse geometry binary predicate list of length 1220, where the predicate was `intersects`  
## first 10 elements:  
## 1: 1  
## 2: 1  
## 3: (empty)  
## 4: 1  
## 5: (empty)  
## 6: (empty)  
## 7: (empty)  
## 8: (empty)  
## 9: 1  
## 10: (empty)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial - interno

```
# filtro espacial - interno
rc_nas_floresta_int <- rc_nas %>%
  dplyr::filter(sf::st_intersects(x = ., y = rc_cob_floresta, sparse = FALSE))
rc_nas_floresta_int
```

```
## Simple feature collection with 169 features and 5 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: 217622.9 ymin: 7505568 xmax: 246181.4 ymax: 7537185
## Projected CRS: SIRGAS 2000 / UTM zone 23S
## First 10 features:
##   GEOCODIGO MUNICIPIO UF CD_UF   HIDRO      geometry
## 1   3543907 RIO CLARO SP   35 nascente POINT (217622.9 7528315)
## 2   3543907 RIO CLARO SP   35 nascente POINT (217836.5 7528103)
## 3   3543907 RIO CLARO SP   35 nascente POINT (218288.9 7528237)
## 4   3543907 RIO CLARO SP   35 nascente POINT (218583.5 7528215)
## 5   3543907 RIO CLARO SP   35 nascente POINT (219062.2 7528499)
## 6   3543907 RIO CLARO SP   35 nascente POINT (219246.2 7529011)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial [] - interno

```
# filtro espacial com [] - interno
rc_nas_floresta_int ← rc_nas[rc_cob_floresta, ]
rc_nas_floresta_int
```

```
## Simple feature collection with 169 features and 5 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: 217622.9 ymin: 7505568 xmax: 246181.4 ymax: 7537185
## Projected CRS: SIRGAS 2000 / UTM zone 23S
## First 10 features:
##      GEOCODIGO MUNICIPIO UF CD_UF      HIDRO      geometry
## 1      3543907 RIO CLARO SP      35 nascente POINT (217622.9 7528315)
## 2      3543907 RIO CLARO SP      35 nascente POINT (217836.5 7528103)
## 4      3543907 RIO CLARO SP      35 nascente POINT (218288.9 7528237)
## 9      3543907 RIO CLARO SP      35 nascente POINT (218583.5 7528215)
## 15     3543907 RIO CLARO SP      35 nascente POINT (219062.2 7528499)
## 19     3543907 RIO CLARO SP      35 nascente POINT (219246.2 7529011)
## 23     3543907 RIO CLARO SP      35 nascente POINT (219554.5 7528573)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial [] - interno

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_cob_floresta$geometry, col = "forestgreen", add = TRUE)  
plot(rc_nas_floresta_int$geometry, col = "blue", pch = 20, cex = 1, add = TRUE)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial - externo

```
# filtro espacial - externo
rc_nas_floresta_ext <- rc_nas %>%
  dplyr::filter(sf::st_disjoint(x = ., y = rc_cob_floresta, sparse = FALSE))
rc_nas_floresta_ext
```

```
## Simple feature collection with 1051 features and 5 fields
```

```
## Geometry type: POINT
```

```
## Dimension:      XY
```

```
## Bounding box:  xmin: 217988.9 ymin: 7504132 xmax: 246367.4 ymax: 7537855
```

```
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

```
## First 10 features:
```

```
##      GEOCODIGO MUNICIPIO UF CD_UF      HIDRO      geometry
## 1      3543907 RIO CLARO SP      35 nascente POINT (217988.9 7528203)
## 2      3543907 RIO CLARO SP      35 nascente POINT (218346.6 7530583)
## 3      3543907 RIO CLARO SP      35 nascente POINT (218393.1 7528031)
## 4      3543907 RIO CLARO SP      35 nascente POINT (218508.3 7528470)
## 5      3543907 RIO CLARO SP      35 nascente POINT (218535.4 7530642)
## 6      3543907 RIO CLARO SP      35 nascente POINT (218760.1 7530258)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial [] - externo

```
# filtro espacial com [] - externo
rc_nas_floresta_ext ← rc_nas[rc_cob_floresta, , op = st_disjoint]
rc_nas_floresta_ext
```

```
## Simple feature collection with 1051 features and 5 fields
## Geometry type: POINT
## Dimension:      XY
## Bounding box:   xmin: 217988.9 ymin: 7504132 xmax: 246367.4 ymax: 7537855
## Projected CRS: SIRGAS 2000 / UTM zone 23S
## First 10 features:
##   GEOCODIGO MUNICIPIO UF CD_UF   HIDRO      geometry
## 3   3543907 RIO CLARO SP   35 nascente POINT (217988.9 7528203)
## 5   3543907 RIO CLARO SP   35 nascente POINT (218346.6 7530583)
## 6   3543907 RIO CLARO SP   35 nascente POINT (218393.1 7528031)
## 7   3543907 RIO CLARO SP   35 nascente POINT (218508.3 7528470)
## 8   3543907 RIO CLARO SP   35 nascente POINT (218535.4 7530642)
## 10  3543907 RIO CLARO SP   35 nascente POINT (218760.1 7530258)
## 11  3543907 RIO CLARO SP   35 nascente POINT (218821.1 7529750)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.1 Filtro espacial

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_cob_floresta$geometry, col = "forestgreen", add = TRUE)  
plot(rc_nas_floresta_ext$geometry, col = "steelblue", pch = 20, cex = 1, add = TRUE)
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.2 Junção espacial

```
# juncao espacial
rc_nas_cob_jun <- rc_nas %>%
  sf::st_join(x = ., y = rc_cob) %>%
  dplyr::group_by(CLASSE_USO) %>%
  dplyr::summarise(n = n())
rc_nas_cob_jun
```

```
## Simple feature collection with 5 features and 2 fields
## Geometry type: MULTIPOINT
## Dimension:      XY
## Bounding box:   xmin: 217622.9 ymin: 7504132 xmax: 246367.4 ymax: 7537855
## Projected CRS: SIRGAS 2000 / UTM zone 23S
## # A tibble: 5 × 3
##   CLASSE_USO      n geomet
##   <chr>      <int> <list>
## 1 água          7 ((224989.9 7527331), (226855.6 7518588), (226859.8 7518617), (227604.7 751627
## 2 área antropizada 4 ((217988.9 7528203), (218346.6 7530583), (218393.1 7528031), (21850
## 3 área edificada  15 ((230717 7522791), (231255.1 7520519), (231420.5 7520332), (232278.7 7516953),
```

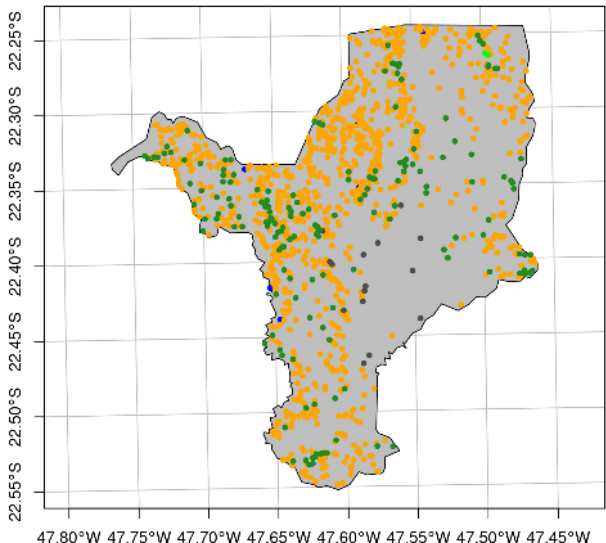


# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.2 Junção espacial

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_nas_cob_jun[1], col = c("blue", "orange", "gray30", "forestgreen", "green"), pch = 20, add = TRUE)
```



# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.3 Agregação espacial

```
# agregacao espacial
rc_cob_nas_agre ← rc_nas %>%
  aggregate(x = ., by = rc_cob, FUN = length)
rc_cob_nas_agre
```

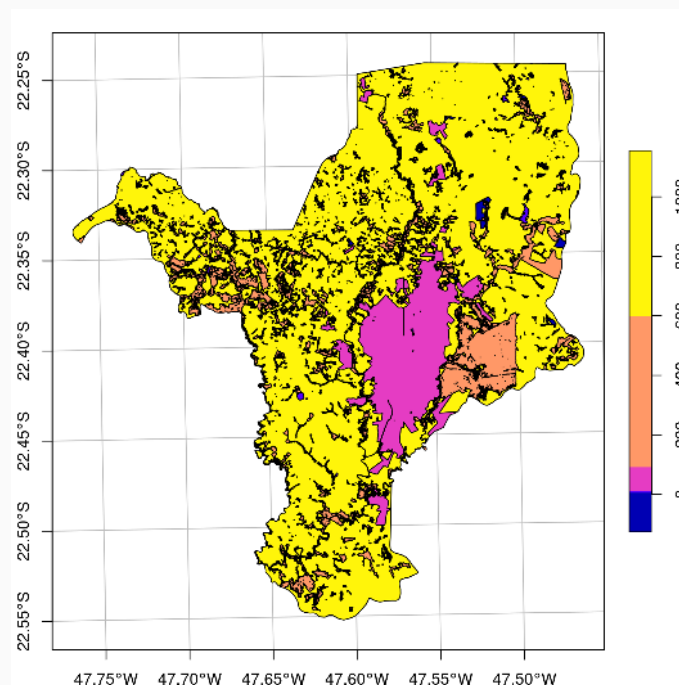
```
## Simple feature collection with 5 features and 5 fields
## Geometry type: MULTIPOLYGON
## Dimension:      XY
## Bounding box:   xmin: 215151.7 ymin: 7503723 xmax: 246582.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO   UF CD_UF HIDRO geometry
## 1         7         7    7     7     7 MULTIPOLYGON (((235487.6 75 ...
## 2        1027        1027 1027  1027  1027 MULTIPOLYGON (((232275 7504 ...
## 3         15         15   15    15    15 MULTIPOLYGON (((233123.6 75 ...
## 4        169        169  169   169   169 MULTIPOLYGON (((232355 7504 ...
## 5         2         2    2     2     2 MULTIPOLYGON (((243052.1 75 ...
```

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.3 Agregação espacial

```
# plot  
plot(rc_cob_nas_agre[1], axes = TRUE, graticule = TRUE, main = NA)
```



# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.4 Distância espacial

```
# distancia
rc_nas_dist_flo <- rc_nas %>%
  dplyr::mutate(dist_flo = sf::st_distance(rc_nas, rc_cob_floresta))
rc_nas_dist_flo
```

```
## Simple feature collection with 1220 features and 6 fields
```

```
## Geometry type: POINT
```

```
## Dimension:      XY
```

```
## Bounding box:  xmin: 217622.9 ymin: 7504132 xmax: 246367.4 ymax: 7537855
```

```
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

```
## First 10 features:
```

##	GEOCODIGO	MUNICIPIO	UF	CD_UF	HIDRO	geometry	dist_flo
## 1	3543907	RIO CLARO	SP	35	nascente	POINT (217622.9 7528315)	0.00000 [m]
## 2	3543907	RIO CLARO	SP	35	nascente	POINT (217836.5 7528103)	0.00000 [m]
## 3	3543907	RIO CLARO	SP	35	nascente	POINT (217988.9 7528203)	37.41369 [m]
## 4	3543907	RIO CLARO	SP	35	nascente	POINT (218288.9 7528237)	0.00000 [m]
## 5	3543907	RIO CLARO	SP	35	nascente	POINT (218346.6 7530583)	714.91131 [m]
## 6	3543907	RIO CLARO	SP	35	nascente	POINT (218393.1 7528031)	21.87610 [m]

# 8. Operações com objetos sf

## 2 Operações espaciais

### 2.4 Distância espacial

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_cob_floresta$geometry, col = "forestgreen", add = TRUE)
plot(rc_nas_dist_flo[7], pch = 20, add = TRUE)
```

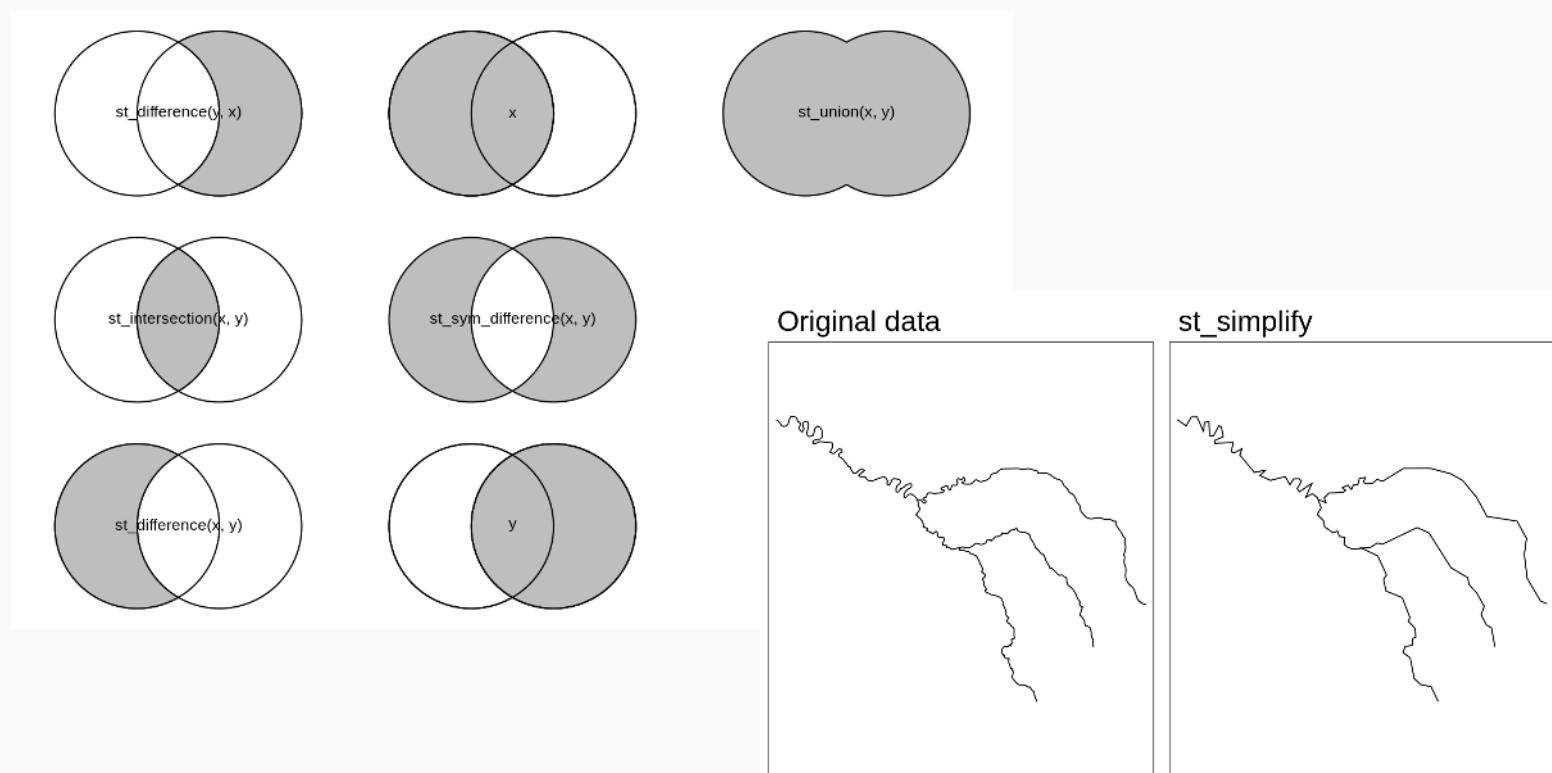
# 8. Operações com objetos sf

## 3 Operações geométricas

Modificação de objetos espaciais baseado em operações que mudam a **geometria do vetor**

### Operações

1. Simplificação
2. Centroides
3. Pontos aleatórios
4. Buffers
5. Polígono convexo
6. Polígonos de Voronoi
7. Quadrículas e hexágonos
8. União ("dissolver")
9. Recorte ("clipar")
10. Transformações de tipo



# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.1 Simplificação

```
# simplificacao
rc_hid_simplificado ← sf::st_simplify(x = rc_hid, dTolerance = 1000)
rc_hid_simplificado
```

```
## Simple feature collection with 1 feature and 6 fields
## Geometry type: MULTILINESTRING
## Dimension:      XY
## Bounding box:   xmin: 215155.3 ymin: 7504132 xmax: 246367.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          HIDRO COMP_KM          geometry
## 1   3543907 RIO CLARO SP      35 curso d'água (0 - 10m) 1142.98 MULTILINESTRING ((231815.7 ...
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.1 Simplificação

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_hid$geometry, col = "steelblue", lwd = 2, add = TRUE)  
plot(rc_hid_simplificado$geometry, col = adjustcolor("black", .7), add = TRUE)
```



# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.2 Centroides

```
# centroides
rc_2020_sirgas2000_utm23s_cent ← sf::st_centroid(rc_2020_sirgas2000_utm23s)
rc_2020_sirgas2000_utm23s_cent
```

```
## Simple feature collection with 1 feature and 7 fields
```

```
## Geometry type: POINT
```

```
## Dimension:      XY
```

```
## Bounding box:   xmin: 234338.3 ymin: 7523516 xmax: 234338.3 ymax: 7523516
```

```
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

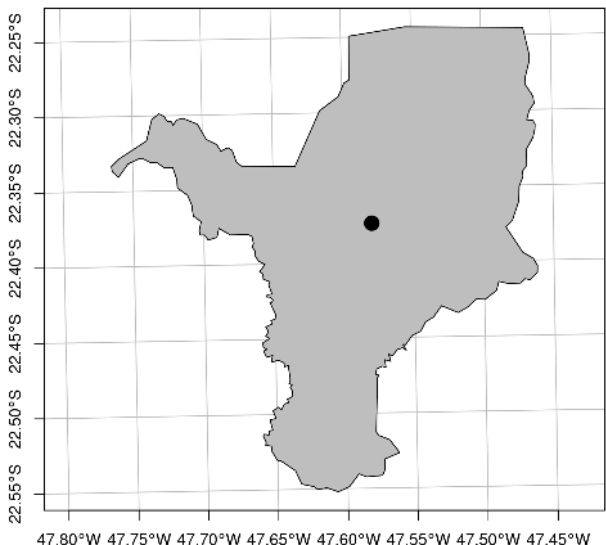
```
##      code_muni name_muni code_state abbrev_state name_state code_region name_region      geom
## 493    3543907 Rio Claro          35           SP  São Paulo           3      Sudeste POINT (234338.3 7523516)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.2 Centroides

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_2020_sirgas2000_utm23s_cent$geom, cex = 3, pch = 20, add = TRUE)
```



# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.3 Pontos aleatórios

```
# fixar amostragem
set.seed(42)

# pontos aleatorios
rc_2020_sirgas2000_utm23s_pontos_aleatorios <- sf::st_sample(rc_2020_sirgas2000_utm23s, size = 30)
rc_2020_sirgas2000_utm23s_pontos_aleatorios
```

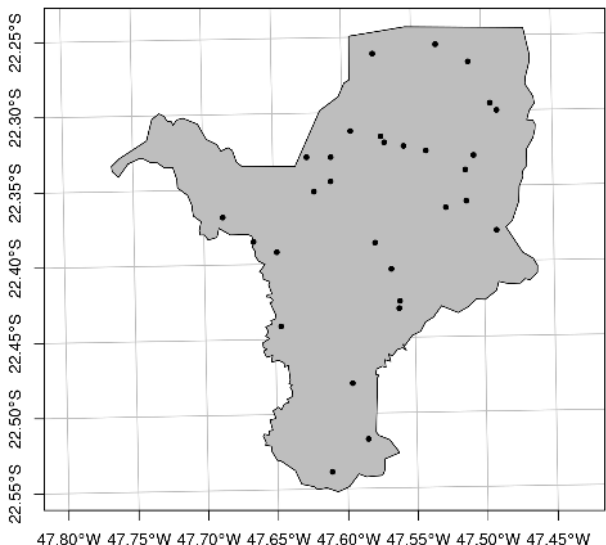
```
## Geometry set for 30 features
## Geometry type: POINT
## Dimension:      XY
## Bounding box:  xmin: 223360.4 ymin: 7505201 xmax: 243540.8 ymax: 7536696
## Projected CRS: SIRGAS 2000 / UTM zone 23S
## First 5 geometries:
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.3 Pontos aleatórios

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios, pch = 20, add = TRUE)
```



# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.4 Buffer

```
# buffer  
rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer ← sf::st_buffer(x = rc_2020_sirgas2000_utm23s_pontos_aleator  
rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer
```

```
## Geometry set for 30 features
```

```
## Geometry type: POLYGON
```

```
## Dimension:      XY
```

```
## Bounding box:  xmin: 222360.4 ymin: 7504201 xmax: 244540.8 ymax: 7537696
```

```
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

```
## First 5 geometries:
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.4 Buffer

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer, col = NA, lwd = 2, border = "red", add = TRUE)  
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios, pch = 20, cex = 1, add = TRUE)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.5 Polígono convexo

```
# poligono convexo
rc_2020_sirgas2000_utm23s_convexo ← rc_2020_sirgas2000_utm23s_pontos_aleatorios %>%
  sf::st_union() %>%
  sf::st_convex_hull()
rc_2020_sirgas2000_utm23s_convexo
```

```
## Geometry set for 1 feature
## Geometry type: POLYGON
## Dimension:      XY
## Bounding box:   xmin: 223360.4 ymin: 7505201 xmax: 243540.8 ymax: 7536696
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.5 Polígono convexo

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_2020_sirgas2000_utm23s_convexo, col = NA, lwd = 2, border = "red", add = TRUE)  
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios, pch = 20, cex = 1, add = TRUE)
```



# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.6 Polígonos de Voronoi

```
# poligono de voronoi
rc_2020_sirgas2000_utm23s_voronoi ← rc_2020_sirgas2000_utm23s_pontos_aleatorios %>%
  sf::st_union() %>%
  sf::st_voronoi()
rc_2020_sirgas2000_utm23s_voronoi
```

```
## Geometry set for 1 feature
## Geometry type: GEOMETRYCOLLECTION
## Dimension:      XY
## Bounding box:   xmin: 191865.2 ymin: 7473706 xmax: 275036.1 ymax: 7568192
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.6 Polígonos de Voronoi

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_2020_sirgas2000_utm23s_voronoi, col = NA, lwd = 2, border = "red", add = TRUE)  
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios, pch = 20, cex = 1, add = TRUE)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.7 Quadrículas e hexágonos

```
# quadriculas
rc_2020_sirgas2000_utm23s_grid <- sf::st_make_grid(x = rc_2020_sirgas2000_utm23s, cellsize = 2000, what = "polygons")
rc_2020_sirgas2000_utm23s_grid <- sf::st_as_sf(rc_2020_sirgas2000_utm23s_grid) %>%
  dplyr::filter(sf::st_intersects(x = ., y = rc_2020_sirgas2000_utm23s, sparse = FALSE))

# centroides das quadriculas
rc_2020_sirgas2000_utm23s_grid_cent <- rc_2020_sirgas2000_utm23s %>%
  sf::st_make_grid(cellsize = 2000, what = "centers") %>%
  sf::st_as_sf() %>%
  dplyr::filter(sf::st_intersects(x = ., y = sf::st_union(rc_2020_sirgas2000_utm23s_grid), sparse = FALSE))
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.7 Quadrículas e hexágonos

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_2020_sirgas2000_utm23s_grid, col = NA, border = "red", lwd = 2, add = TRUE)
plot(rc_2020_sirgas2000_utm23s_grid_cent, pch = 20, add = TRUE)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.7 Quadrículas e hexágonos

```
# hexagonos
rc_2020_sirgas2000_utm23s_hex <- rc_2020_sirgas2000_utm23s %>%
  sf::st_make_grid(cellsize = 2000, square = FALSE) %>%
  sf::st_as_sf() %>%
  dplyr::filter(sf::st_intersects(x = ., y = rc_2020_sirgas2000_utm23s, sparse = FALSE))

# centroides de hexagonos
rc_2020_sirgas2000_utm23s_hex_cent <- rc_2020_sirgas2000_utm23s %>%
  sf::st_make_grid(cellsize = 2000, square = FALSE, what = "centers") %>%
  sf::st_as_sf() %>%
  dplyr::filter(sf::st_intersects(x = ., y = sf::st_union(rc_2020_sirgas2000_utm23s_hex), sparse = FALSE))
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.7 Quadrículas e hexágonos

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_2020_sirgas2000_utm23s_hex, col = NA, border = "red", lwd = 2, add = TRUE)  
plot(rc_2020_sirgas2000_utm23s_hex_cent, pch = 20, add = TRUE)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.8 União ("dissolver")

```
# uniao
rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uniao ← sf::st_union(rc_2020_sirgas2000_utm23s_pontos_aleato
rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uniao
```

```
## Geometry set for 1 feature
```

```
## Geometry type: MULTIPOLYGON
```

```
## Dimension:      XY
```

```
## Bounding box:   xmin: 222360.4 ymin: 7504201 xmax: 244540.8 ymax: 7537696
```

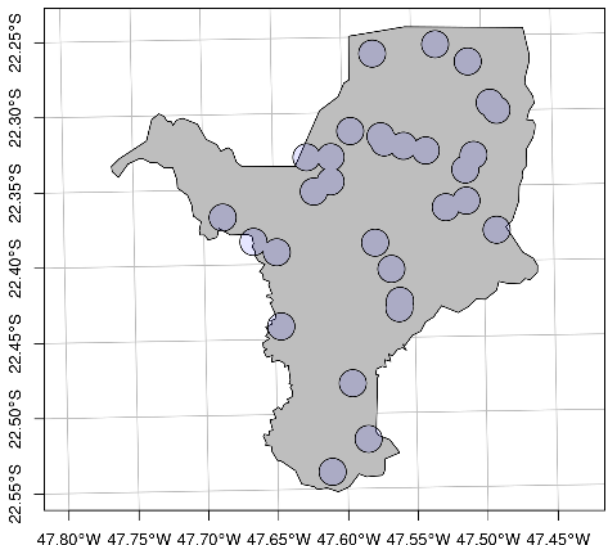
```
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.8 União ("dissolver")

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uniao, col = adjustcolor("blue", .1), add = TRUE)
```





# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.9 Recorte ("clipar")

```
# recorte - interseccao
rc_hid_interseccao ← sf::st_intersection(x = rc_hid, y = rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uni
rc_hid_interseccao
```

```
## Simple feature collection with 1 feature and 6 fields
## Geometry type: MULTILINESTRING
## Dimension:      XY
## Bounding box:   xmin: 222421.9 ymin: 7504269 xmax: 244527.4 ymax: 7537655
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          HIDRO COMP_KM          geometry
## 1   3543907 RIO CLARO SP     35 curso d'água (0 - 10m) 1142.98 MULTILINESTRING ((231820 75 ...
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.9 Recorte ("clipar")

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uniao, col = adjustcolor("blue", .1), add = TRUE)  
plot(rc_hid_interseccao$geometry, col = "blue", add = TRUE)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.9 Recorte ("clipar") - diferença

```
# recorte - diferenca
rc_hid_diferenca ← sf::st_difference(x = rc_hid, y = rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uniao)
rc_hid_diferenca
```

```
## Simple feature collection with 1 feature and 6 fields
## Geometry type: MULTILINESTRING
## Dimension:      XY
## Bounding box:   xmin: 215155.3 ymin: 7504132 xmax: 246367.4 ymax: 7537978
## Projected CRS: SIRGAS 2000 / UTM zone 23S
##   GEOCODIGO MUNICIPIO UF CD_UF          HIDRO COMP_KM          geometry
## 1   3543907 RIO CLARO SP     35 curso d'água (0 - 10m) 1142.98 MULTILINESTRING ((231815.7 ...
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.9 Recorte ("clipar") - diferença

```
# plot  
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)  
plot(rc_2020_sirgas2000_utm23s_pontos_aleatorios_buffer_uniao, col = adjustcolor("blue", .1), add = TRUE)  
plot(rc_hid_diferenca$geometry, col = "blue", add = TRUE)
```

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.10 Transformações de tipo

```
# transformacao de tipo
rc_cob_floresta_polygon <- rc_cob_floresta %>%
  sf::st_cast("POLYGON") %>%
  dplyr::mutate(area_ha = sf::st_area(.)/1e4 %>% round(2))
rc_cob_floresta_polygon
```

```
## Simple feature collection with 1732 features and 7 fields
```

```
## Geometry type: POLYGON
```

```
## Dimension: XY
```

```
## Bounding box: xmin: 215442.4 ymin: 7504235 xmax: 246282.3 ymax: 7537969
```

```
## Projected CRS: SIRGAS 2000 / UTM zone 23S
```

```
## First 10 features:
```

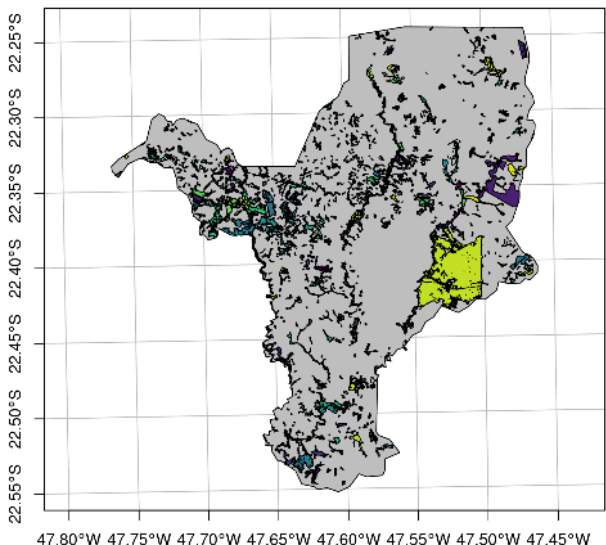
##	GEOCODIGO	MUNICIPIO	UF	CD_UF	CLASSE_USO	AREA_HA	geometry	area_ha
## 1	3543907	RIO CLARO	SP	35	formação florestal	7017.99	POLYGON ((232355 7504440, 2 ...	3.232500e+00 [m^2]
## 1.1	3543907	RIO CLARO	SP	35	formação florestal	7017.99	POLYGON ((229470 7505200, 2 ...	7.015000e+00 [m^2]
## 1.2	3543907	RIO CLARO	SP	35	formação florestal	7017.99	POLYGON ((231185 7505305, 2 ...	2.992500e+00 [m^2]
## 1.3	3543907	RIO CLARO	SP	35	formação florestal	7017.99	POLYGON ((232175 7505460, 2 ...	3.750002e-01 [m^2]
## 1.4	3543907	RIO CLARO	SP	35	formação florestal	7017.99	POLYGON ((231720 7505435, 2 ...	5.017500e+00 [m^2]

# 8. Operações com objetos sf

## 3 Operações geométricas

### 3.10 Transformações de tipo

```
# plot
plot(rc_2020_sirgas2000_utm23s$geom, col = "gray", main = NA, axes = TRUE, graticule = TRUE)
plot(rc_cob_floresta_polygon["area_ha"], col = viridis::viridis(100), add = TRUE)
```



# 9. Exportar objetos sf

## Exportar como shapefile

```
# exportar o vetor de floresta na extensão esri shapefile  
sf::st_write(obj = rc_cob_floresta_polygon, dsn = here::here("03_dados", "vetor", "rc_cob_floresta_polygon.shp"))
```



SHP



DBF



SHX



PRJ



# 9. Exportar objetos sf

## Exportar como geopackage

```
# exportar o vetor de floresta na extensão geopackage  
sf::st_write(obj = rc_cob_floresta_polygon, dsn = here::here("03_dados", "vetor", "vetores.gpkg"), layer = "rc_cob_floresta_polygon")
```



[Switch from Shapefile](#), [Using the Geopackage Format with R](#)



# 9. Exportar objetos sf

## Exportar como geopackage

```
# exportar o vetor da america do sul na extensão geopackage  
sf::st_write(obj = sa, dsn = here::here("03_dados", "vetor", "vetores.gpkg"), layer = "south_america")
```



[Switch from Shapefile](#), [Using the Geopackage Format with R](#)

Dúvidas?

# Maurício Vancine

Contatos:

✉ [mauricio.vancine@gmail.com](mailto:mauricio.vancine@gmail.com)

🐦 [@mauriciovancine](https://twitter.com/mauriciovancine)

🌀 [mauriciovancine](https://github.com/mauriciovancine)

🔗 [mauriciovancine.github.io](https://mauriciovancine.github.io)



Slides criados via pacote [xaringan](https://github.com/josephmccaskey/xaringan) e tema [Metropolis](https://github.com/josephmccaskey/metropolis). Animação dos sapos por [@probzz](https://twitter.com/probzz).