Introduction to the `sf` Package

Dr. Arun Mitra

Introduction

- **Objective**: Learn the basics of the `sf` package in R for spatial data analysis.
- **Why `sf`?**: Simplifies handling, analysis, and visualization of spatial data in R.

Overview of Spatial Data in R

- **Spatial Data**: Data associated with locations in a geometric space.
- **Types**:
  - Point data
  - Line data
  - Polygon data
- **Applications**: Environmental monitoring, urban planning, epidemiology.

The `sf` Package

The **sf** package is an R implementation of Simple Features.

This package incorporates:

- A new spatial data class system in R
- Functions for reading and writing data
- Tools for spatial operations on vectors

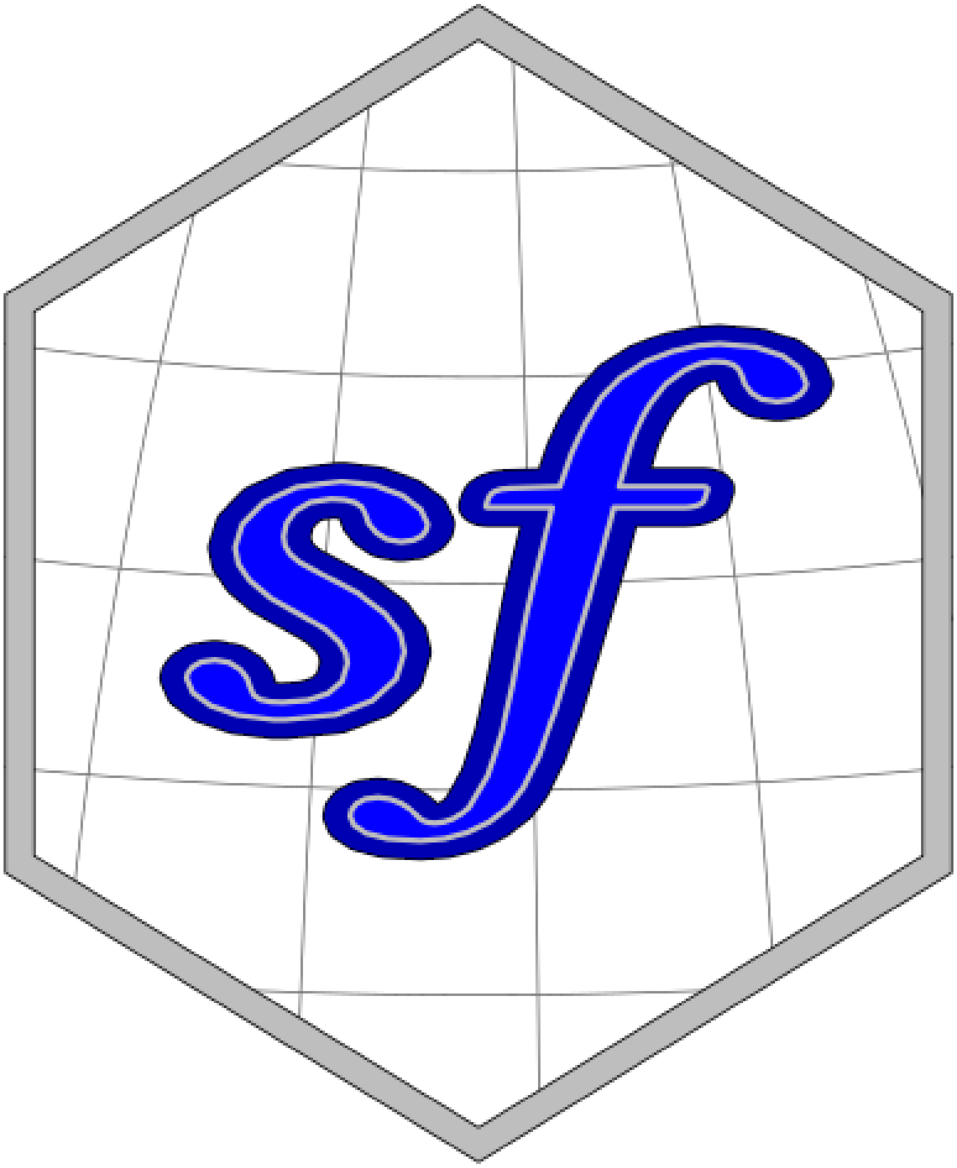


Figure 1: sf package gif

```r
install.packages("sf")
```

Why the `sf` Package?

- **Integration**: Seamlessly integrates with the tidyverse.
- **Efficiency**: More efficient and user-friendly than previous spatial packages.

- **Standards**: Adheres to international standards for spatial data.

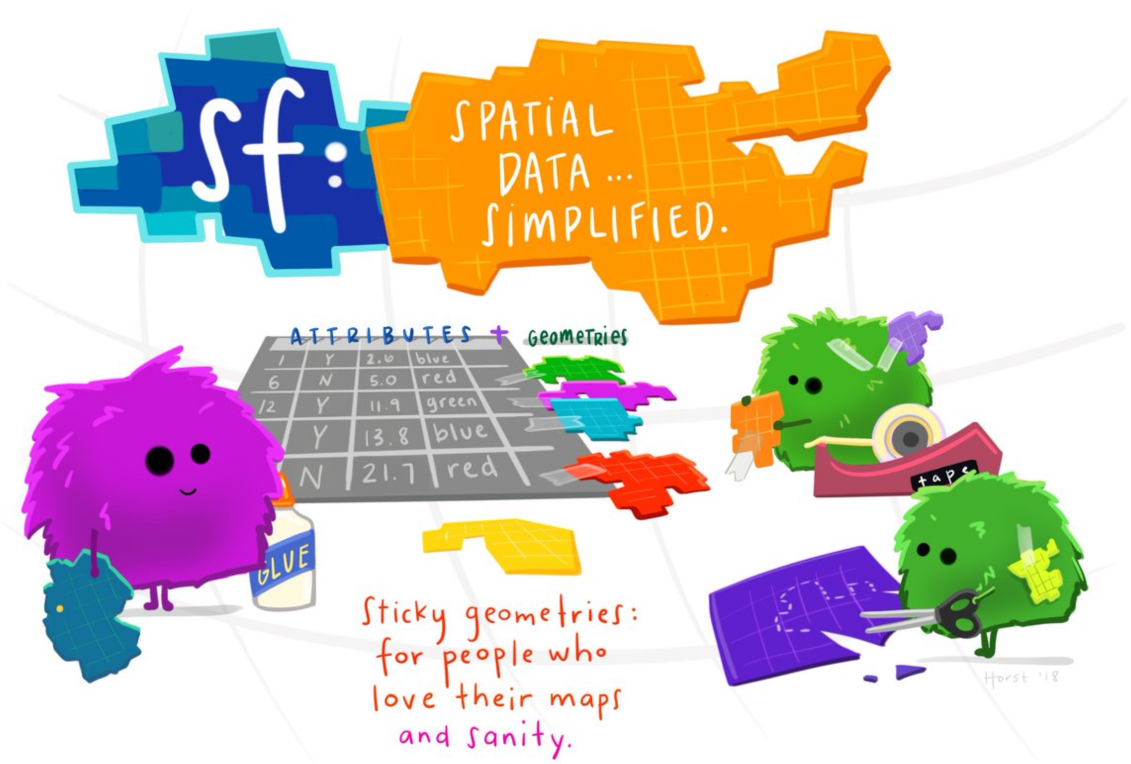
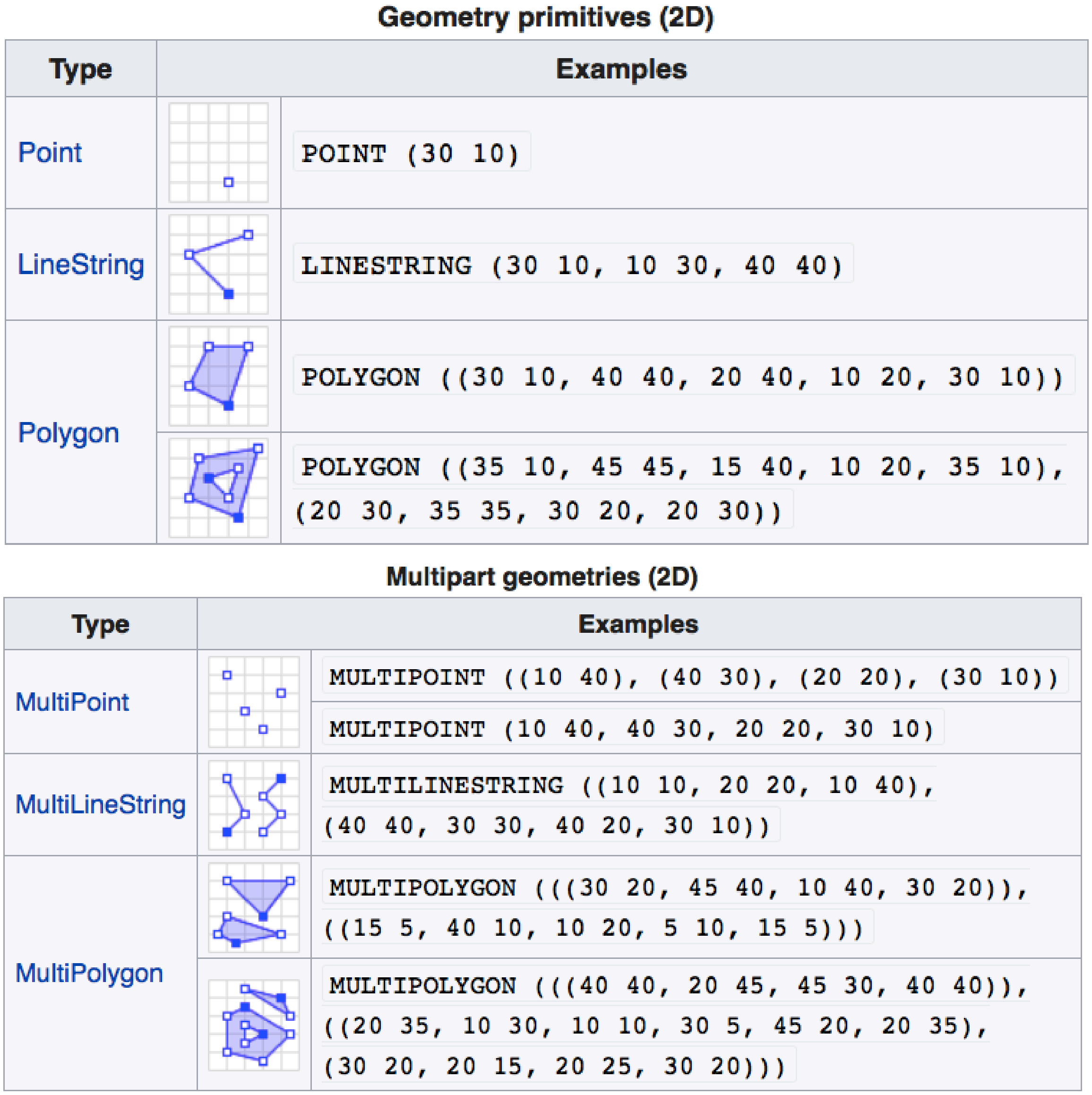


Figure 2: sf package usage

Geometry Types in `sf`



Geometry primitives (2D)

Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`

		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10),`
		`(20 30, 35 35, 30 20, 20 30))`

Multipart geometries (2D)

Type	Examples	
MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`

MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40),`
		`(40 40, 30 30, 40 20, 30 10))`

MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)),`
		`((15 5, 40 10, 10 20, 5 10, 15 5)))`

		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)),`
		`((20 35, 10 30, 10 10, 30 5, 45 20, 20 35),`
		`(30 20, 20 15, 20 25, 30 20)))`

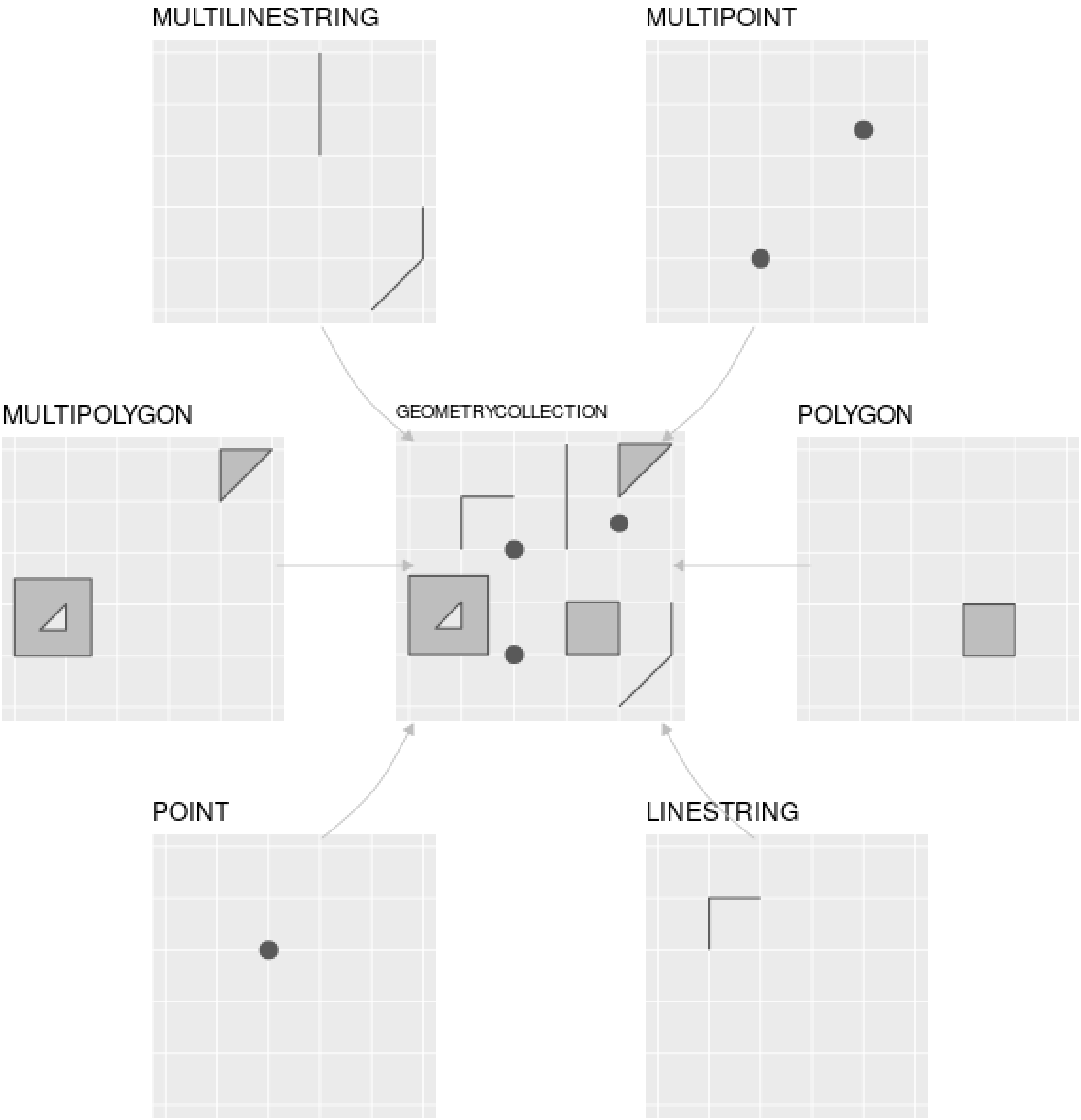


Figure 3: sf Classes

Loading Spatial Data into R using `sf`

```r
library(sf)
path_to_shape_file <- "path/to/shapefile.shp"
spatial_data <- st_read(path_to_shape_file)
```

Viewing the sf Object

```
print(spatial_data)
```

Plotting the sf Object

```
ggplot(spatial_data) +
  geom_sf()
```

```
ggplot(spatial_data) +
  geom_sf(aes(color = some_attribute))
```

Concept of the sf Package

- **Spatial Data Frame**: Combines attributes and geometry.
- **Key Functions**:
  - `st_read()`: Read spatial data.
  - `st_write()`: Write spatial data.
  - `st_transform()`: Transform coordinate systems.

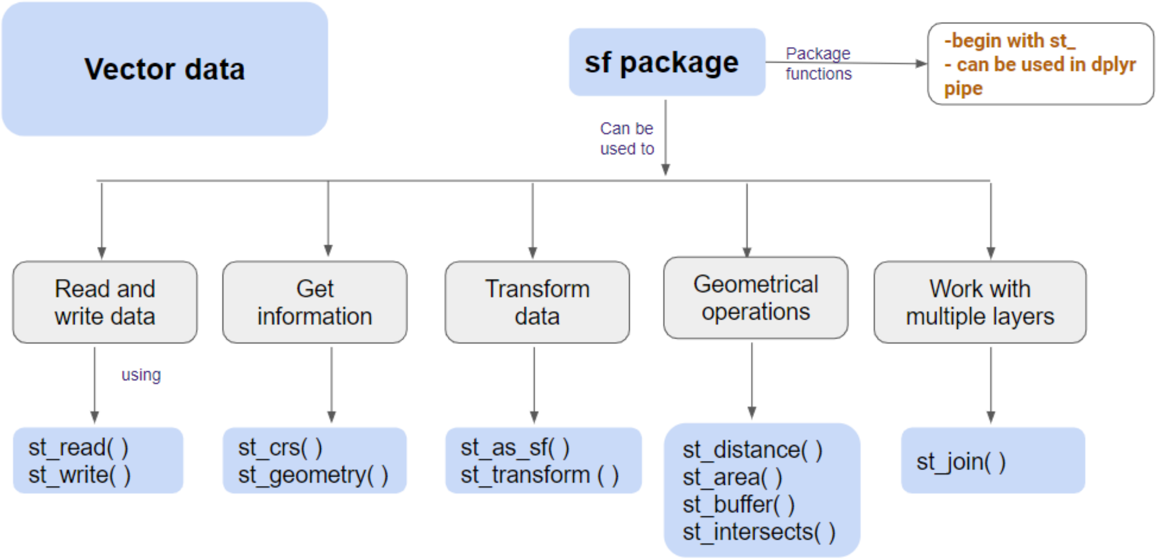


Figure 4: sf Concept Map

Dependencies of the `sf` Package

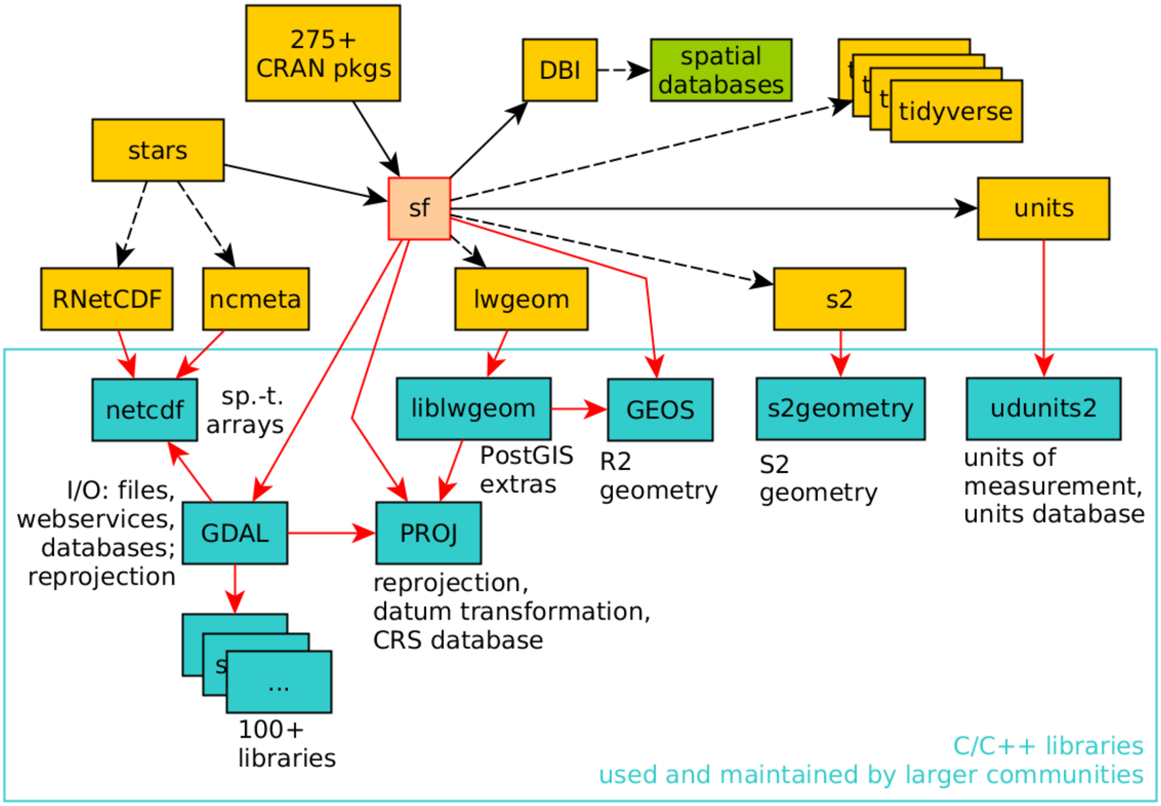


Figure 5: sf Dependencies

- **Key Dependencies**:
  - GDAL: Geospatial Data Abstraction Library
  - PROJ: Cartographic Projections Library
  - GEOS: Geometry Engine

Methods in `sf`

```
methods(class="sf")
```

- **Common Methods**:
  - `st_union()`: Union of geometries.
  - `st_intersection()`: Intersection of geometries.
  - `st_buffer()`: Buffer around geometries.

Interactive Mapping with `sf`

```r
library(mapview)
mapview(spatial_data)
```

Practical Exercise: Loading and Plotting Data

1. **Load Data**:
   - Use `st_read()` to load spatial data.
   - Example shapefile: `"path/to/shapefile.shp"`
2. **View Data**:
   - Print the `sf` object.
3. **Plot Data**:
   - Use `ggplot2` to create a basic map.

```r
library(sf)
spatial_data <- st_read("path/to/shapefile.shp")
print(spatial_data)
ggplot(spatial_data) + geom_sf()
```

Practical Exercise: Advanced Plotting

1. **Color by Attribute**:
   - Use `aes()` to map colors to an attribute.
2. **Interactive Map**:
   - Use `mapview` for interactive mapping.

```r
ggplot(spatial_data) + geom_sf(aes(color = attribute))
library(mapview)
mapview(spatial_data)
```

Spatial Operations with `sf`

- **Buffering**: Create buffer zones around geometries.

```r
buffered <- st_buffer(spatial_data, dist = 100)
ggplot(buffered) + geom_sf()
```

- **Intersection**: Find intersecting areas between geometries.

```r
intersection <- st_intersection(spatial_data, another_spatial_layer)
ggplot(intersection) + geom_sf()
```

Spatial Joins with `sf`

- **Spatial Join**: Combine attributes from different spatial datasets based on their spatial relationship.

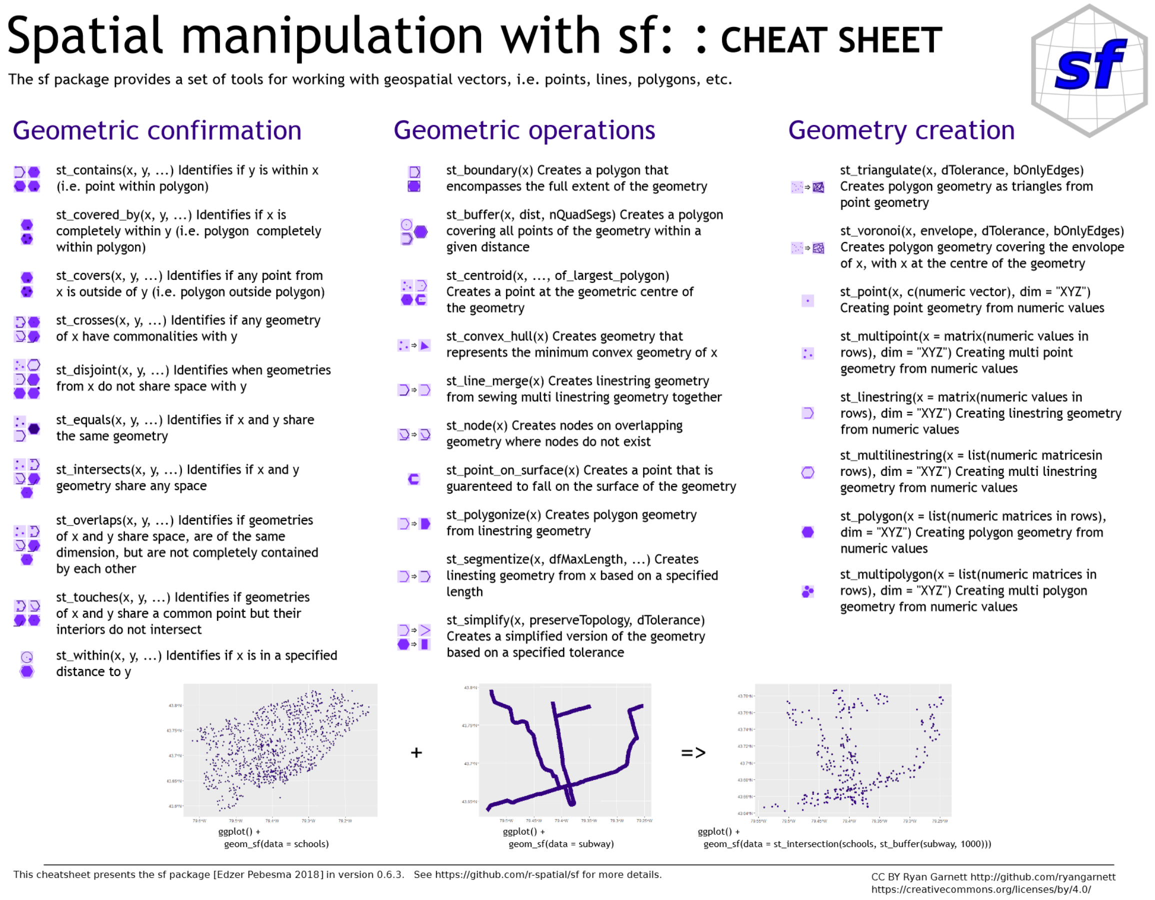
```r
joined_data <- st_join(spatial_data, another_spatial_layer)
ggplot(joined_data) + geom_sf()
```

Coordinate Transformations with `sf`

- **Transform Coordinates**: Change the coordinate reference system (CRS) of spatial data.

```r
transformed_data <- st_transform(spatial_data, crs = 4326)
ggplot(transformed_data) + geom_sf()
```

Where to Look for Help?



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 y is within x (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

```
ggplot() +
  geom_sf(data = schools)
```

+

```
ggplot() +
  geom_sf(data = subway)
```

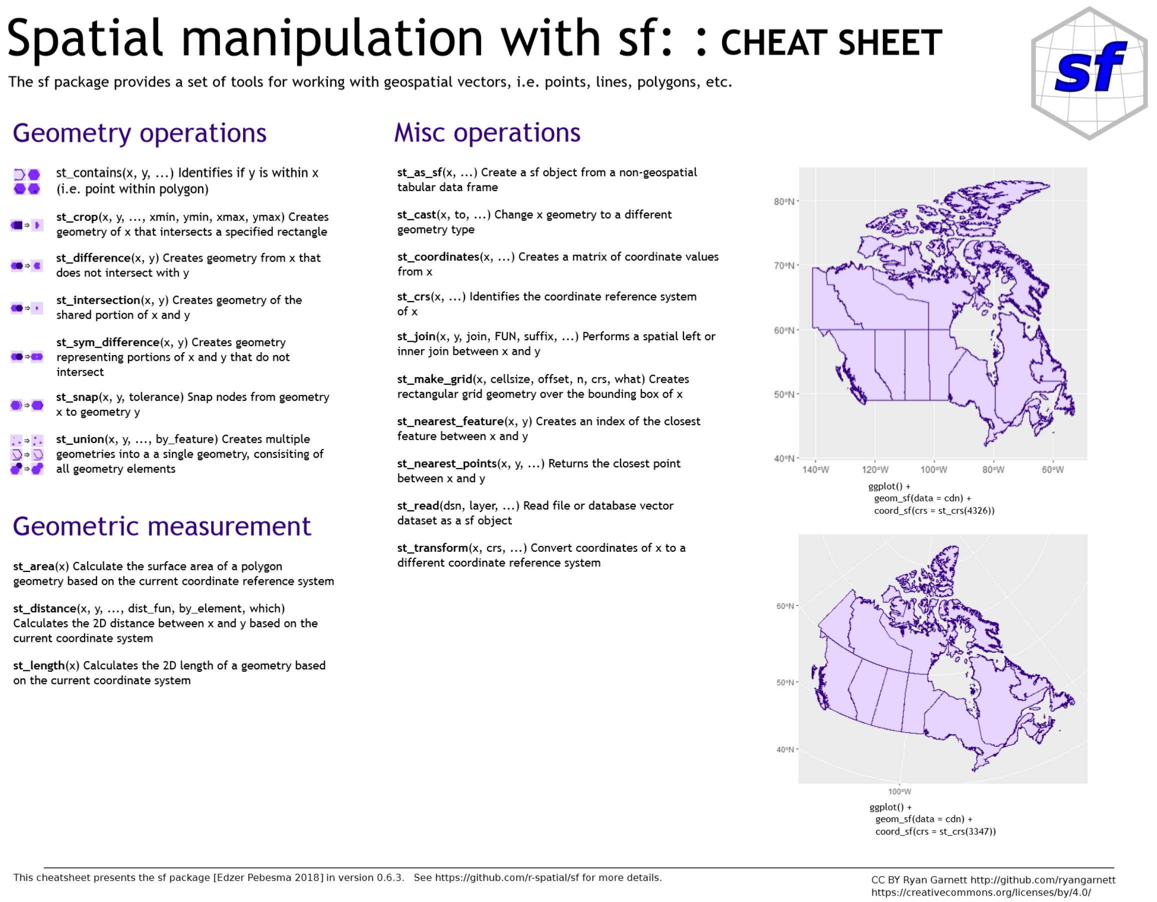
=>

```
ggplot() +
  geom_sf(data = st_intersection(schools, st_buffer(subway, 1000)))
```

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

CC BY Ryan Garnett http://github.com/ryangarnett
https://creativecommons.org/licenses/by/4.0/

Figure 6: sf Cheatsheet 1



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.

Geometry operations

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

st_crop(x, y, ..., xmin, ymin, xmax, ymax) Creates geometry of x that intersects a specified rectangle

st_difference(x, y) Creates geometry from x that does not intersect with y

st_intersection(x, y) Creates geometry of the shared portion of x and y

st_sym_difference(x, y) Creates geometry representing portions of x and y that do not intersect

st_snap(x, y, tolerance) Snap nodes from geometry x to geometry y

st_union(x, y, ..., by_feature) Creates multiple geometries into a a single geometry, consisiting of all geometry elements

Geometric measurement

st_area(x) Calculate the surface area of a polygon geometry based on the current coordinate reference system

st_distance(x, y, ..., dist_fun, by_element, which) Calculates the 2D distance between x and y based on the current coordinate system

st_length(x) Calculates the 2D length of a geometry based on the current coordinate system

Misc operations

st_as_sf(x, ...) Create a sf object from a non-geospatial tabular data frame

st_cast(x, to, ...) Change x geometry to a different geometry type

st_coordinates(x, ...) Creates a matrix of coordinate values from x

st_crs(x, ...) Identifies the coordinate reference system of x

st_join(x, y, join, FUN, suffix, ...) Performs a spatial left or inner join between x and y

st_make_grid(x, cellsize, offset, n, crs, what) Creates rectangular grid geometry over the bounding box of x

st_nearest_feature(x, y) Creates an index of the closest feature between x and y

st_nearest_points(x, y, ...) Returns the closest point between x and y

st_read(dsn, layer, ...) Read file or database vector dataset as a sf object

st_transform(x, crs, ...) Convert coordinates of x to a different coordinate reference system

```
ggplot() +
  geom_sf(data = cdn) +
  coord_sf(crs = st_crs(4326))
```

```
ggplot() +
  geom_sf(data = cdn) +
  coord_sf(crs = st_crs(3347))
```

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 7: sf Cheatsheet 2

- **Resource**: sf Cheatsheet

Questions

- **Any doubts or questions?**
- **Hands-on practice time!**