Lesson 8: Programming GeoVisual Analytics with R

Dr. Kam Tin Seong
Assoc. Professor of Information Systems

School of Computing and Information Systems,
Singapore Management University

2020-2-15 (updated: 2021-06-29)

Content

- Handling Geospatial Data with R
  - Simple features approach: **sf** package
  - Raster package
- Visualising Geospatial Data with R
  - Popular R packages for visualising geospatial data
  - Getting to know **tmap** package
  - Working with **tmap**

Analysis of Spatial Data

Base R includes many functions that can be used for reading, vizualising, and analysing spatial data. The focus in this view is on "geographical" spatial...[more]

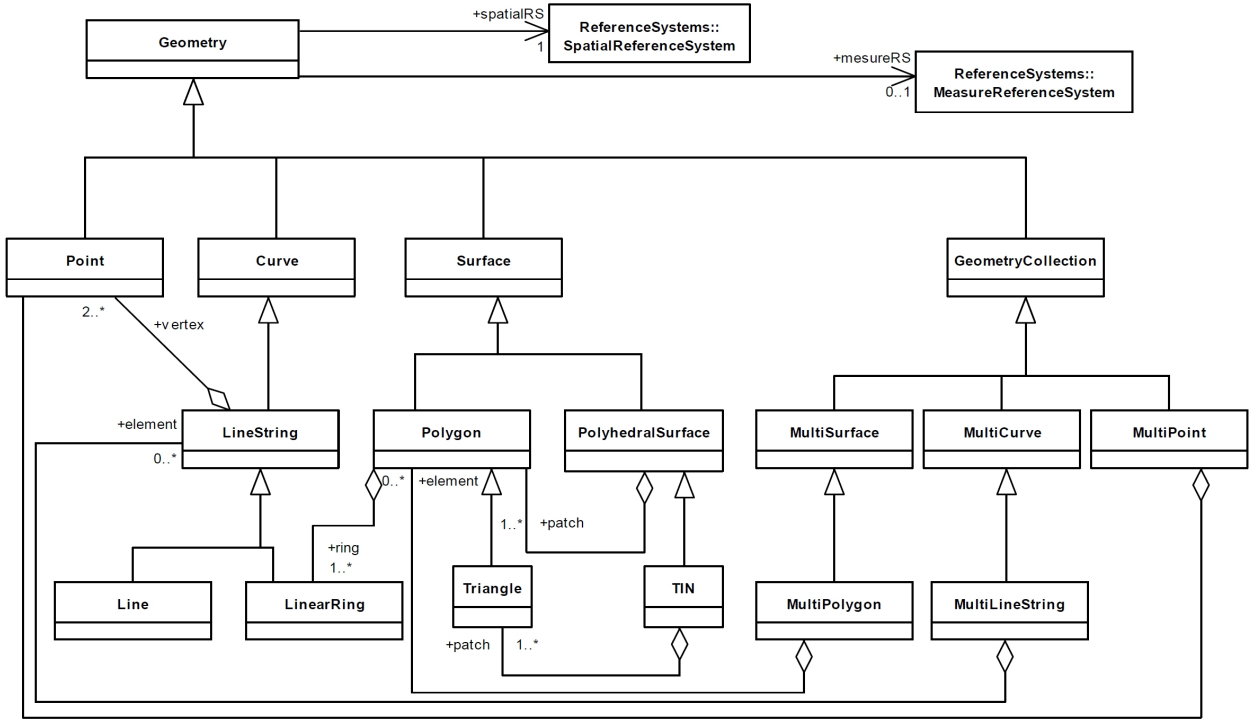
Click here

An introduction to Simple Features

- **feature**: abstraction of real world phenomena (type or instance); has a geometry and other attributes (properties)
- **simple feature**: feature with all geometric attributes described piecewise by straight line or planar interpolation between sets of points (no curves)
- It is a hierarchical data model that simplifies geographic data by condensing a complex range of geographic forms into a single geometry class.

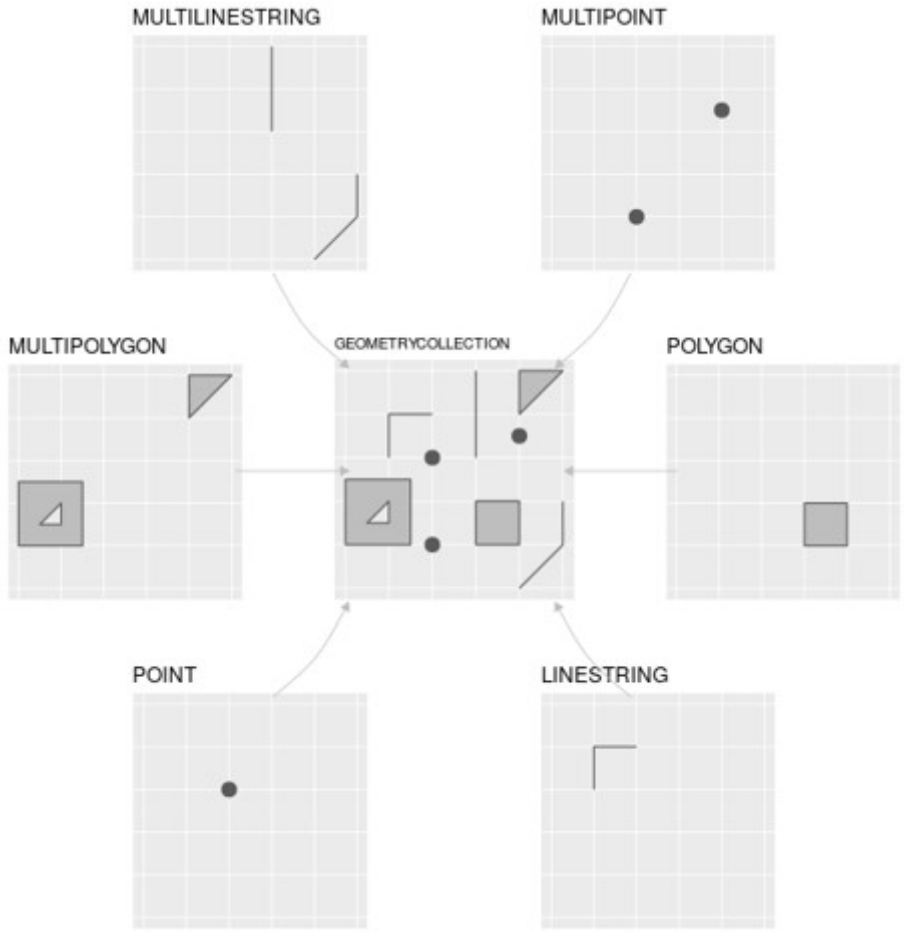
Simple Features Objects

- *Simple features specification* is an open standard developed and endorsed by the Open Geospatial Consortium (OGC) to represent a wide range of geographic information.



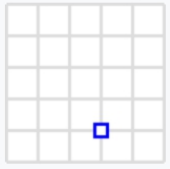
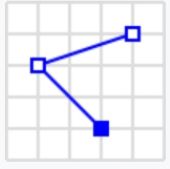
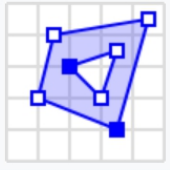
An Introduction to Simple Features

- Only 7 out of 17 possible types of simple feature are currently used in the vast majority of GIS operations.



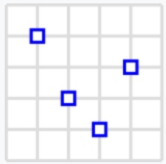
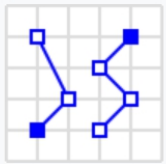
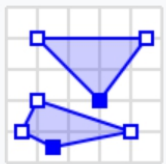
Working with Simple Features

Simple Features: How they look like?

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))`

Working with Simple Features

Simple Features: How they look like?

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)))`

Introducing sf package

Introducing sf Package

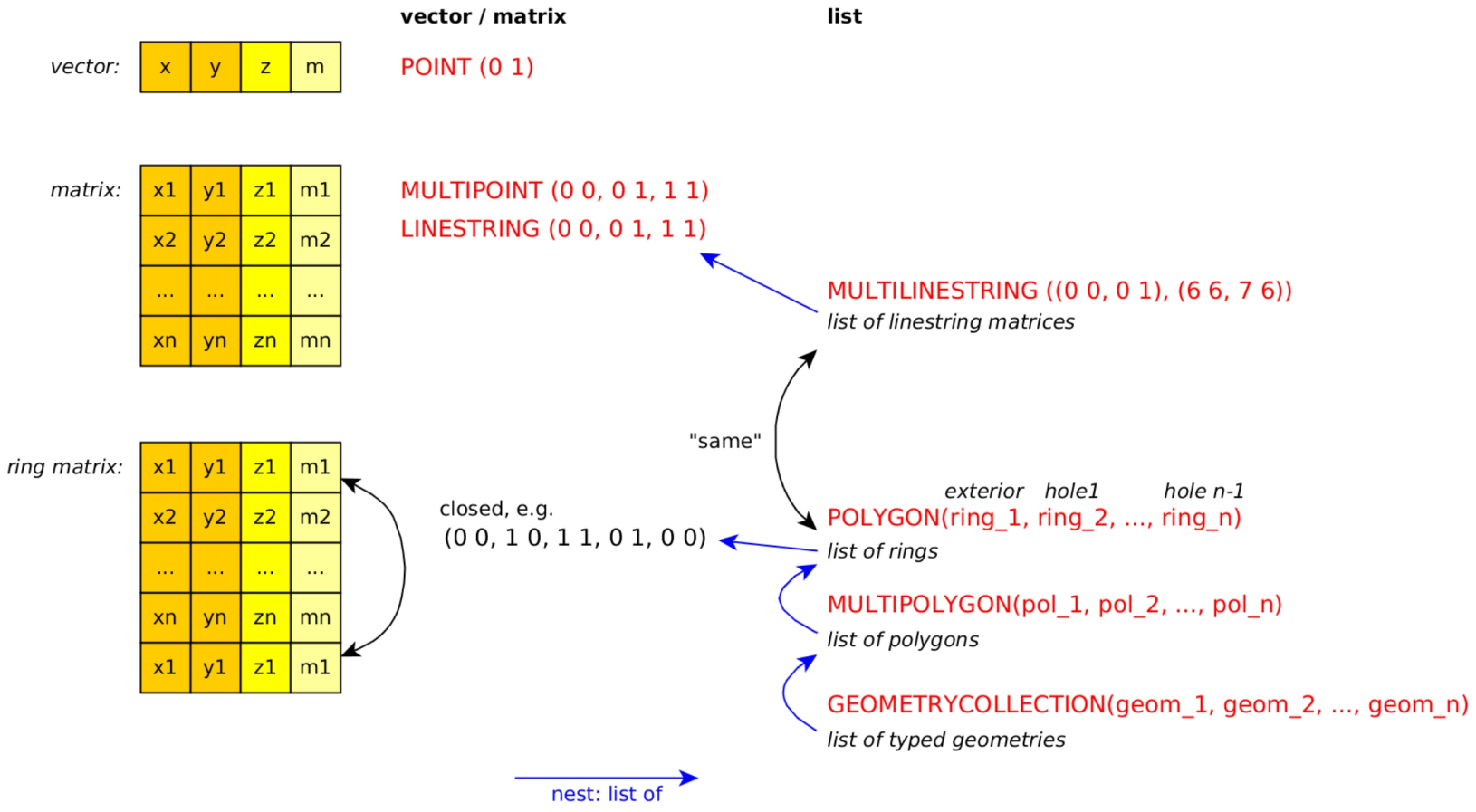
An overview

A package that provides simple features access for R. Package sf:

- represents simple features as records in a data.frame or tibble with a geometry list-column.
- represents natively in R all 17 simple feature types for all dimensions (XY, XYZ, XYM, XYZM).
- interfaces to GEOS to support geometrical operations including the DE9-IM.
- interfaces to GDAL, supporting all driver options, Date and POSIXct and list-columns.
- interfaces to PRØJ for coordinate reference system conversions and transformations.
- uses well-known-binary serialisations written in C++/Rcpp for fast I/O with GDAL and GEOS.
- reads from and writes to spatial databases such as PostGIS using DBI.
- is extended by pkg lwgeom for further liblwgeom/PostGIS functions, including some spherical geometry functions.

Introducing sf package

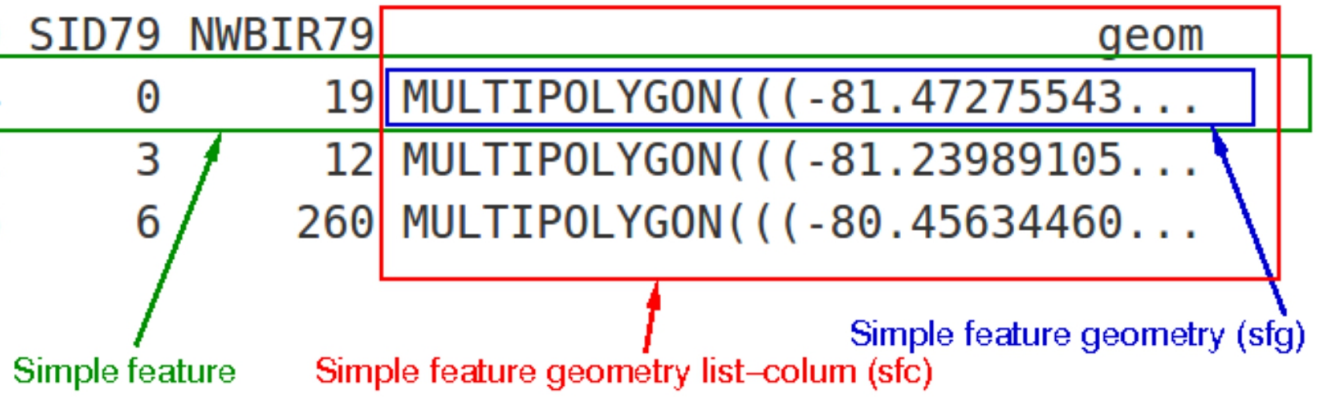
sfg : geometry for one feature



Introducing sf package

sf: objects with simple features

```
## 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...
```



Introducing sf package

sf: objects with simple features

- in green a **simple feature**: a single record, or data.frame row, consisting of attributes and geometry.
- in blue a single **simple feature geometry** (an object of class **sfg**).
- in red a **simple feature list-column** (an object of class **sfc**, which is a column in the data.frame).

Introducing sf package

Precision

One of the attributes of a geometry list-column (sfc) is the precision: a double number that, when non-zero, causes some rounding during conversion to WKB, which might help certain geometrical operations succeed that would otherwise fail due to floating point representation.

The model is that of GEOS, which copies from the Java Topology Suite (JTS), and works like this:

- if precision is zero (default, unspecified), nothing is modified.
- negative values convert to float (4-byte real) precision.
- positive values convert to round(x*precision)/precision.

sf Methods

Book keeping, low-level I/O

- st_read: read simple features from file or database, or retrieve layer names and their geometry type(s)
- st_as_text: convert to WKT
- st_as_binary: convert to WKB
- st_as_sfc: convert geometries to sfc (e.g., from WKT, WKB) as(x, "Spatial"): convert to Spatial*
- st_as_sf: convert to sf (e.g., convert from Spatial*)

sf Methods

Logical binary geometry predicates

- st_intersects: touch or overlap
- st_disjoint: !intersects
- st_touches: touch
- st_crosses: cross (don't touch)
- st_within: within
- st_contains: contains
- st_overlaps: overlaps
- st_covers: cover
- st_covered_by: covered by
- st_equals: equals
- st_equals_exact: equals, with some fuzz returns a sparse (default) or dense logical matrix

sf Methods

Geometry generating logical operators

- st_union: union of several geometries
- st_intersection: intersection of pairs of geometries
- st_difference: difference between pairs of geometries
- st_sym_difference: symmetric difference (xor)

sf Methods

Higher-level operations: summarise, interpolate, aggregate, st_join

- aggregate and summarise use st_union (by default) to group feature geometries
- st_interpolate_aw: area-weighted interpolation, uses st_intersection to interpolate or redistribute attribute values, based on area of overlap:
- st_join uses one of the logical binary geometry predicates (default: st_intersects) to join records in table pairs

sf Methods

Manipulating geometries

- st_line_merge: merges lines
- st_segmentize: adds points to straight lines
- st_voronoi: creates voronoi tesselation
- st_centroid: gives centroid of geometry
- st_convex_hull: creates convex hull of set of points
- st_triangulate: triangulates set of points (not constrained)
- st_polygonize: creates polygon from lines that form a closed ring
- st_simplify: simplifies lines by removing vertices
- st_split: split a polygon given line geometry
- st_buffer: compute a buffer around this geometry/each geometry
- st_make_valid: tries to make an invalid geometry valid (requires lwgeom)
- st_boundary: return the boundary of a geometry

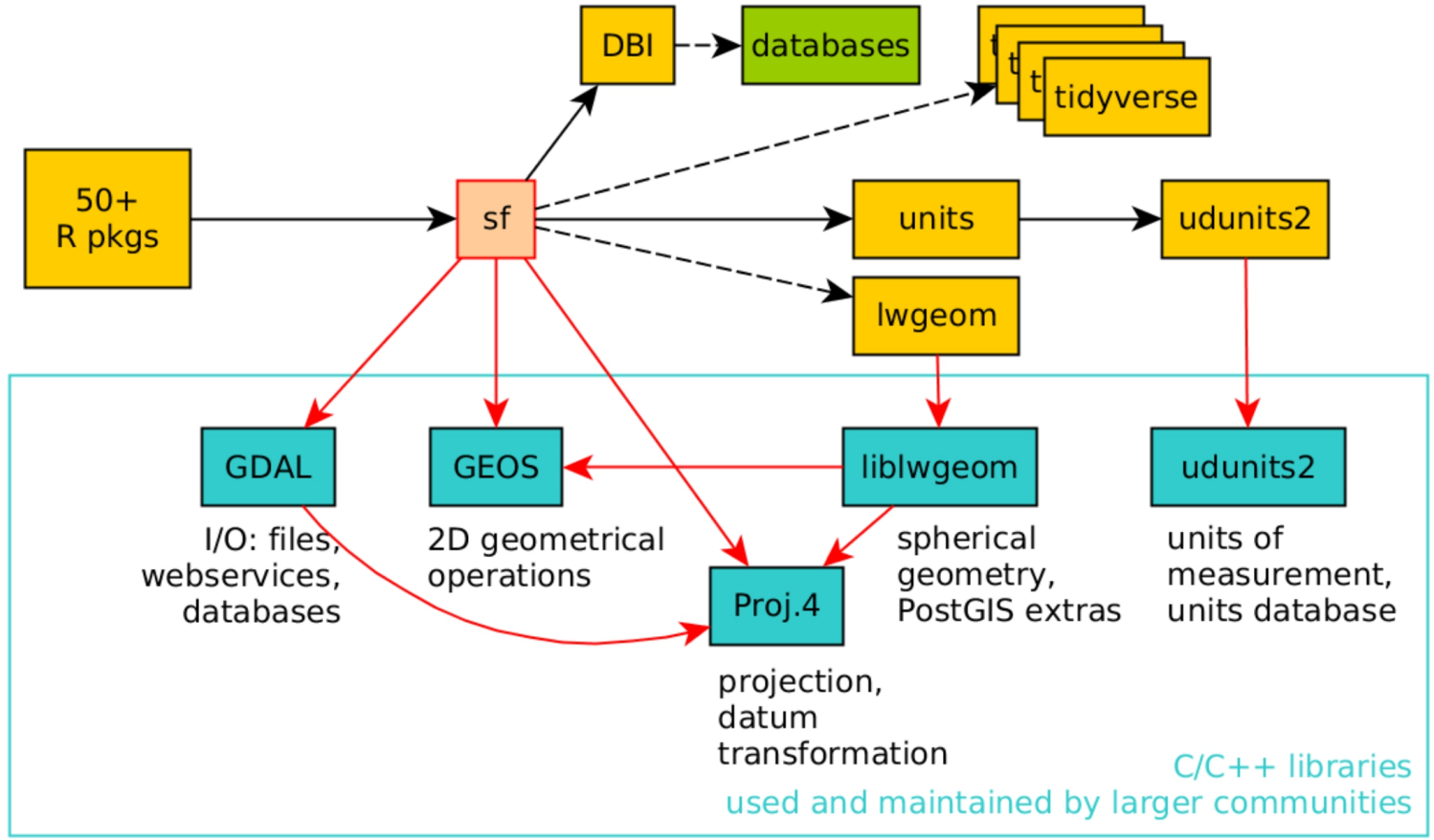
sf Methods

Convenience functions

- st_zm: sets or removes z and/or m geometry
- st_coordinates: retrieve coordinates in a matrix or data.frame
- st_geometry: set, or retrieve sfc from an sf object
- st_is: check whether geometry is of a particular type

Introducing sf package

sf and other geospatial packages



Introducing sf package

sf & tidyverse

- sf spatial objects are data.frames (or tibbles)
- you can always un-sf, and work with tbl_df or data.frame having an sfc list-column
- sf methods for filter, arrange, distinct, group_by, ungroup, mutate, select have sticky geometry
- *st_join()* joins tables based on a spatial predicate
- summarise unions geometry by group (or altogether)

Mapping packages in R

Selected popular mapping packages

CRAN Task View: Analysis of Spatial Data

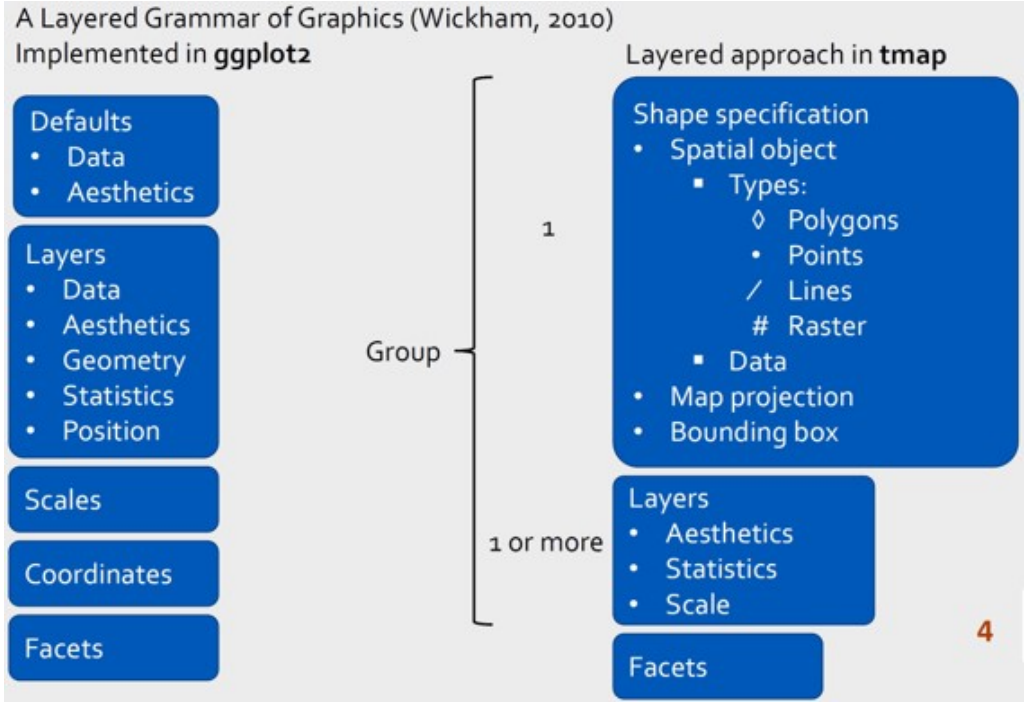
- tmap
- cartography
- leaflet
- ggplot2. Read Chapter 6: Maps of 'ggplot2: Elegant Graphics for Data Analysis' for more detail.
- ggmap
- quickmapr
- mapview

Other packages

- RColorBrewer
- classInt

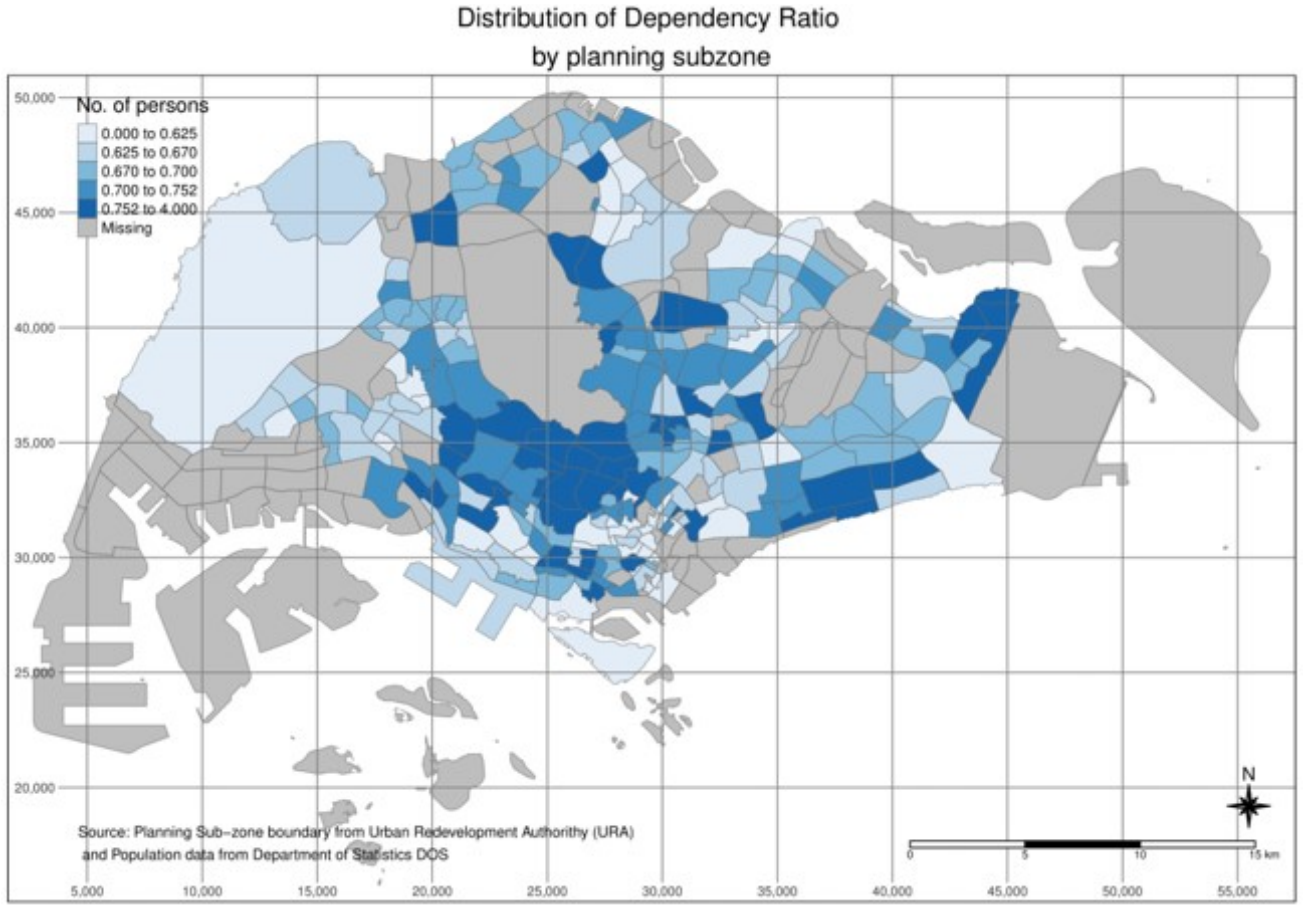
Introducing *tmap*

- *tmap* is a R package specially designed for creating thematic maps using the pricinples of the **Grammar of Graphics**.
- It offers a flexible, layer-based, and easy to use approach to create thematic maps, such as choropleths and proportional symbol maps.



Introducing *tmap*

A choropleth map created using *tmap*



Introducing *tmap*

Under the hood

The R codes used to create the thematic map shown earlier

```
tm_shape(mpsz_agemale2017)+
  tm_fill("DEPENDENCY",
          style = "quantile",
          palette = "Blues",
          title = "No. of persons") +
  tm_layout(main.title = "Distribution of Dependency Ratio \nby
planning subzone",
            main.title.position = "center",
            main.title.size = 1.2,
            legend.height = 0.45,
            legend.width = 0.35,
            frame = TRUE) +
  tm_borders(alpha = 0.5) +
  tm_compass(type="8star", size = 2) +
  tm_scale_bar() +
  tm_grid() +
  tm_credits("Source: Planning Sub-zone boundary from Urban
Redevelopment Authorithy (URA)\n and Population data from
Department of Statistics DOS",
             position = c("left", "bottom"))
```

Introducing *tmap*

Shape objects

- *tmap* supports the class Spatial or Raster, respectively from the *sp* and the *raster* package. The supported subclasses are:

	Without data	With data
Polygons	SpatialPolygons	SpatialPolygonsDataFrame
Points	SpatialPoints	SpatialPointsDataFrame
Lines	SpatialLines	SpatialLinesDataFrame
Raster	SpatialGrid	SpatialGridDataFrame
Raster	SpatialPixels	SpatialPixelsDataFrame
Raster		RasterLayer
Raster		RasterBrick
Raster		RasterStack

- *tmap* also supports **simple features** from the new *sf* package.

Plotting functions of *tmap*

Two approaches can be used to prepare thematic map using *tmap*, they are:

- Plotting a thematic map quickly by using **qtm()**.
- Plotting highly customisable thematic map by using tmap elements.

Plotting functions of *tmap*

qtm()

```
qtm(mpsz_agemale2017, fill = "DEPENDENCY")
```

tmap elements

tm_shape()

- The first element to start with is tm_shape(), which specifies the shape object.

Function	Argument	Description	Spatial types
`tm_shape`	`shp`	shape object	polygons points lines raster cells
	`projection`	projection code	
	`bbox`	bounding box	

tmap elements

Base layers

- Next, one, or a combination of the following drawing layers should be specified:

Drawing layer	Description	Aesthetics
Base layer		
`tm_polygons`	Draw polygons	col
`tm_symbols`	Draws symbols	size, col, shape
`tm_lines`	Draws polylines	col, lwd
`tm_raster`	Draws a raster	col
`tm_text`	Add text labels	text, size, col

tmap element

Base layers

- Each of these functions specifies the geometry, mapping, and scaling component of the LGTM.
- An aesthetic can take a constant value, a data variable name, or a vector consisting of values or variable names.
- If a data variable is provided, the scale is automatically configured according to the values of this variable, but can be adjusted with several arguments. For instance, the main scaling arguments for a color aesthetic are color palette, the preferred number of classes, and a style to create classes.
- Also, for each aesthetic, except for the text labels, a legend is automatically created.
- If a vector of variable names is provided, small multiples are created, which will be explained further below.

tmap elements

Derived layers

- The supported derived layers are as follows:

Derived layer		
`tm_fill`	Fills the polygons	see tm_polygons
`tm_borders`	Draws polygon borders	none
`tm_bubbles`	Draws bubbles	see tm_symbols
`tm_squares`	Draws squares	see tm_symbols
`tm_dots`	Draws dots	see tm_symbols
`tm_markers`	Draws markers	see tm_symbols and tm_text
`tm_iso`	Draws iso/contour lines	see tm_lines and tm_text

tmap elements

Derived layers

- Each aesthetic can take a constant value or a data variable name. For instance, tm_fill(col="blue") colors all polygons blue, while tm_fill(col="var1"), where "var1" is the name of a data variable in the shape object, creates a choropleth.

tmap elements

Attribute layers

- Attribute layers provide support to cartographic furniture such as grid lines, map compass, scale bar and text label.

Function	Description	Main arguments	
`tm_grid`	coordinate grid lines	`x` and `y`	position of grid lines
		`projection`	projection of the grid lines
`tm_credits`	credits text	`text`	credits text
`tm_scale_bar`	scale bar	`breaks`	scale bar breaks
`tm_compass`	map compass	`type`	compass type
`tm_logo`	logo(s)	`file`	filename or URL
`tm_xlab` and `tm_ylab`	axis labels	`text`	label text

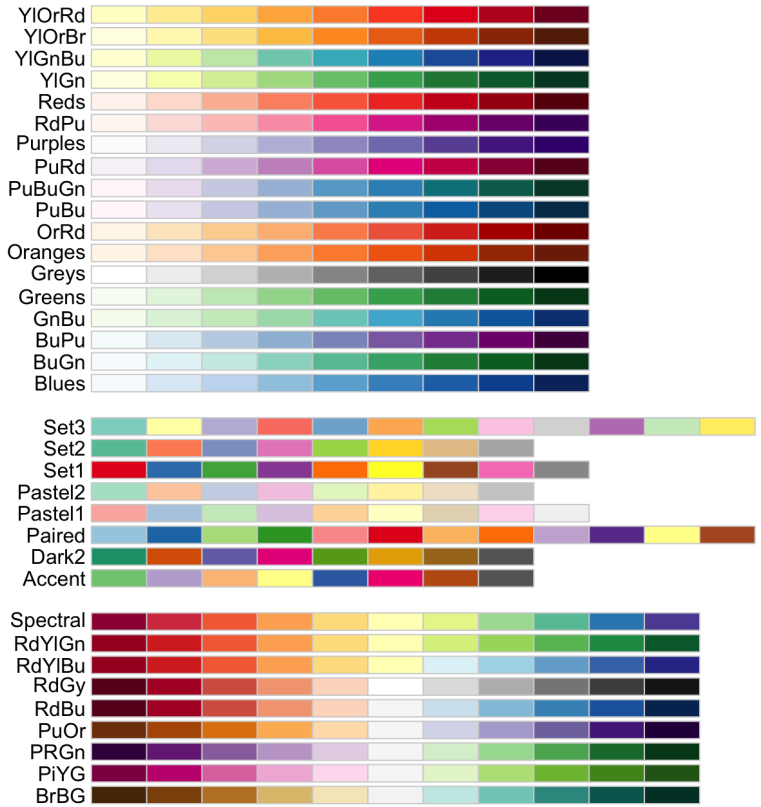
Colour scheme of *tmap*'

The colour schemen

- *tmap* supports colour ramps either defined by the user or a set of predefined colour ramps from the RColorBrewer() function.
- "-" as prefix to reverse the palette.

Colour scheme of *tmap*

RColorBrewer



The code chunk used to retrieve RColorBrewer palette.

```
library(RColorBrewer)
display.brewer.all()
```

Data classification methods of tmap

- The **fixed** style permits a "classIntervals" object to be specified with given breaks, set in the fixedBreaks argument; the length of fixedBreaks should be n+1; this style can be used to insert rounded break values.
- The **sd** style chooses breaks based on pretty of the centred and scaled variables, and may have a number of classes different from n; the returned par= includes the centre and scale values.
- The **equal** style divides the range of the variable into n parts.
- The **pretty** style chooses a number of breaks not necessarily equal to n using pretty, but likely to be legible; arguments to pretty may be passed through.
- The **quantile** style provides quantile breaks; arguments to quantile may be passed through.

Data classification methods of tmap

- The **kmeans** style uses kmeans to generate the breaks; it may be anchored using set.seed; the pars attribute returns the kmeans object generated; if kmeans fails, a jittered input vector containing rtimes replications of var is tried - with few unique values in var, this can prove necessary; arguments to kmeans may be passed through.

- The **hclust** style uses hclust to generate the breaks using hierarchical clustering; the pars attribute returns the hclust object generated, and can be used to find other breaks using getHclustClassIntervals; arguments to hclust may be passed through.

Data classification methods of tmap

- The **bclust** style uses bclust to generate the breaks using bagged clustering; it may be anchored using set.seed; the pars attribute returns the bclust object generated, and can be used to find other breaks using getBclustClassIntervals; if bclust fails, a jittered input vector containing rtimes replications of var is tried - with few unique values in var, this can prove necessary; arguments to bclust may be passed through.

- The **fisher** style uses the algorithm proposed by W. D. Fisher (1958) and discussed by Slocum et. al. (2005) as the Fisher-Jenks algorithm; added here thanks to Hisaji Ono.

- The **jenks** style has been ported from Jenks' Basic code.

Map layout of *tmap*

- The layout of the thematic map can be changed with tm_layout() or one of its wrapper functions.
- The wrapper functions starting with tm*format* specify the format for a specific shape. In the tmap package, a couple of them are included, for instance tm_format_World() that is tailored for world maps. It's also possible to create your own wrapper function for shapes that you will use frequently.
- Besides the shape-dependent tm*format* wrapper(), tmap also contains wrapper functions for shape-independent styles.

Drawing Small Multiples with *tmap*

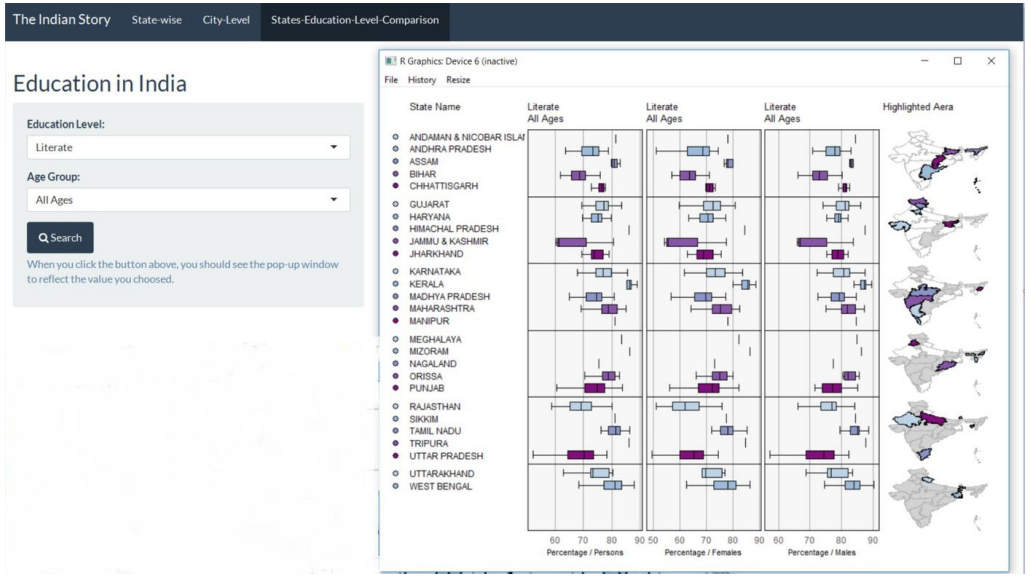
With *tmap*, small multiples can be generated in three ways:

- By assigning multiple values to at least one of the aesthetic arguments
- By defining a group-by variable in tm_facets()
- By creating multiple stand-alone maps with tmap_arrange()

Micromap

- **micromap** is a R package specially designed to display statistical summaries associated with areal units, or polygons.

- The package contains functions, heavily dependent on the utilities of the **ggplot2** package, which may be used to produce a row-oriented graph composed of different panels, or columns, of information. These panels at a minimum contain maps, a legend, and statistical summaries.



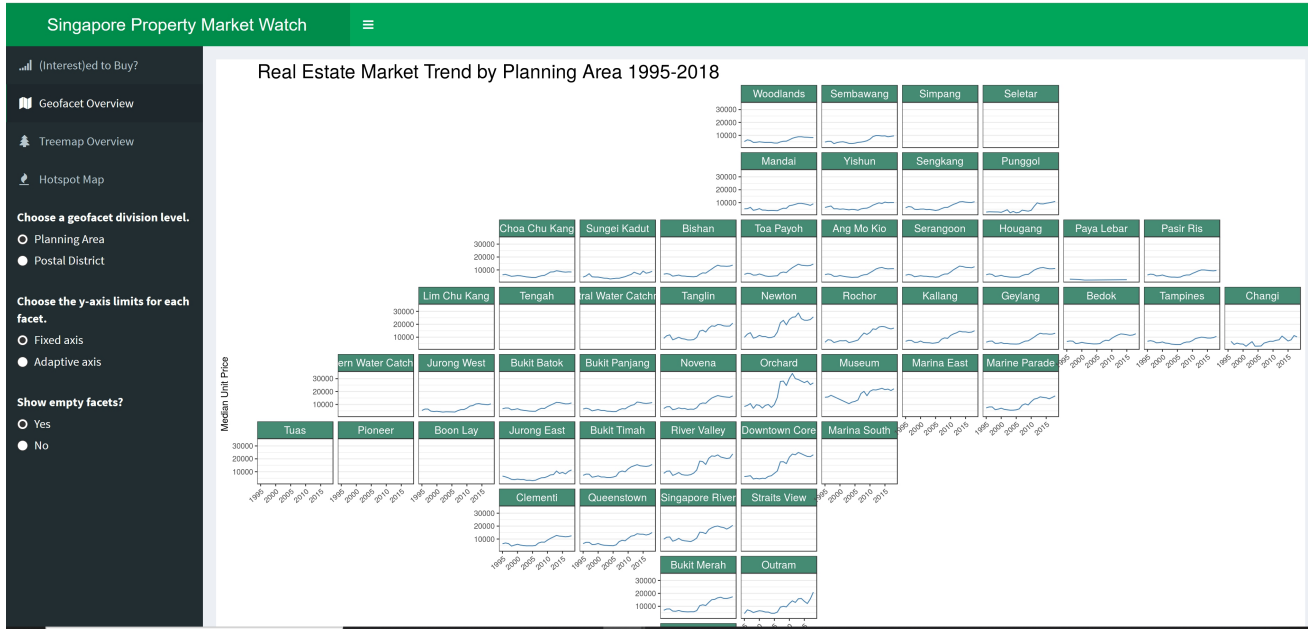
Source: The Indian Story

- This link explains the basic concepts of Interactive Linked Micromap Plots.

- This article provides detail discussion on micromap package.

Geofacet

Geofacet provides geofaceting functionality for 'ggplot2'. Geofaceting arranges a sequence of plots of data for different geographical entities into a grid that preserves some of the geographical orientation.



Source: Singapore Property Market Watch

References

All About sf package

- Reference manual
- Vignettes:
  - 1. Simple Features for R
  - 2. Reading, Writing and Converting Simple Features
  - 3. Manipulating Simple Feature Geometries
  - 4. Manipulating Simple Features
  - 5. Plotting Simple Features
  - 6. Miscellaneous
  - 7. Spherical geometry in sf using s2geometry

All About tmap package

- tmap: Thematic Maps in R
- tmap User Guide
- tmap: get started!
- tmap: changes in version 2.0
- tmap: creating thematic maps in a flexible way (useR!2015)
- Exploring and presenting maps with tmap (useR!2017)