R Programming For Natural Resource Professionals

Lecture 11

Map making in R

Ugly plot contest!

- https://forms.gle/hGYSHFzJoyLqrKjr7

- Select your top 3 choices, in no particular order

The week(s) ahead…

Section III: Statistical applications

Week 12	Apr 18/20	Basic regressions
Week 13	Apr 25/27	Advanced regressions
Week 14	May 2/4	Simulations: resampling/bootstrapping

Assign final homework: Due by midnight on Sun May 15

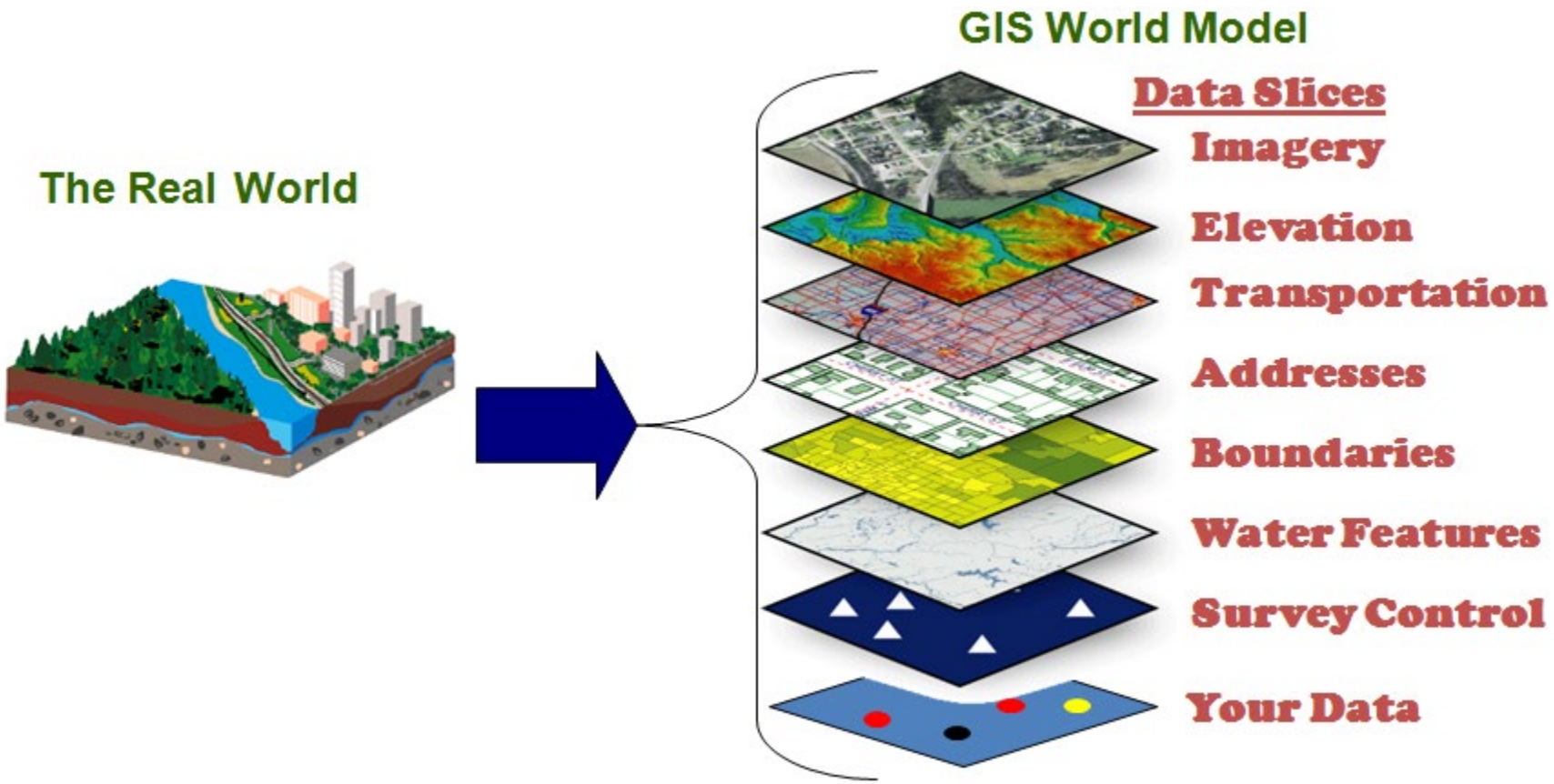
Week 15	May 9/11	Multivariate statistics

Wrapping up last week’s slides

Learning objectives for this week

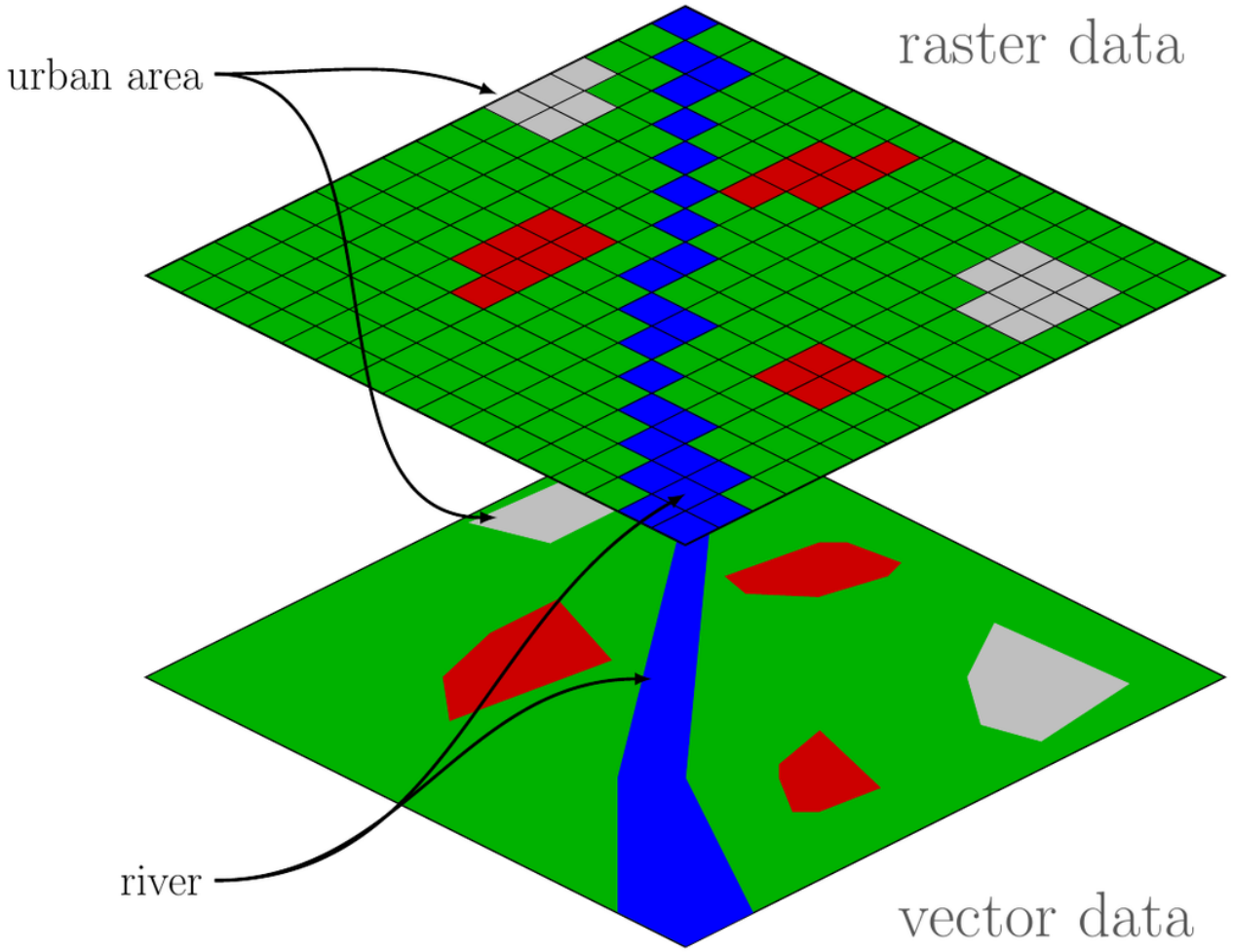
1. Learn some basic GIS concepts
2. Create basic maps in R
3. Incorporate data into maps in R

GIS: Geographic Information System



GIS terms

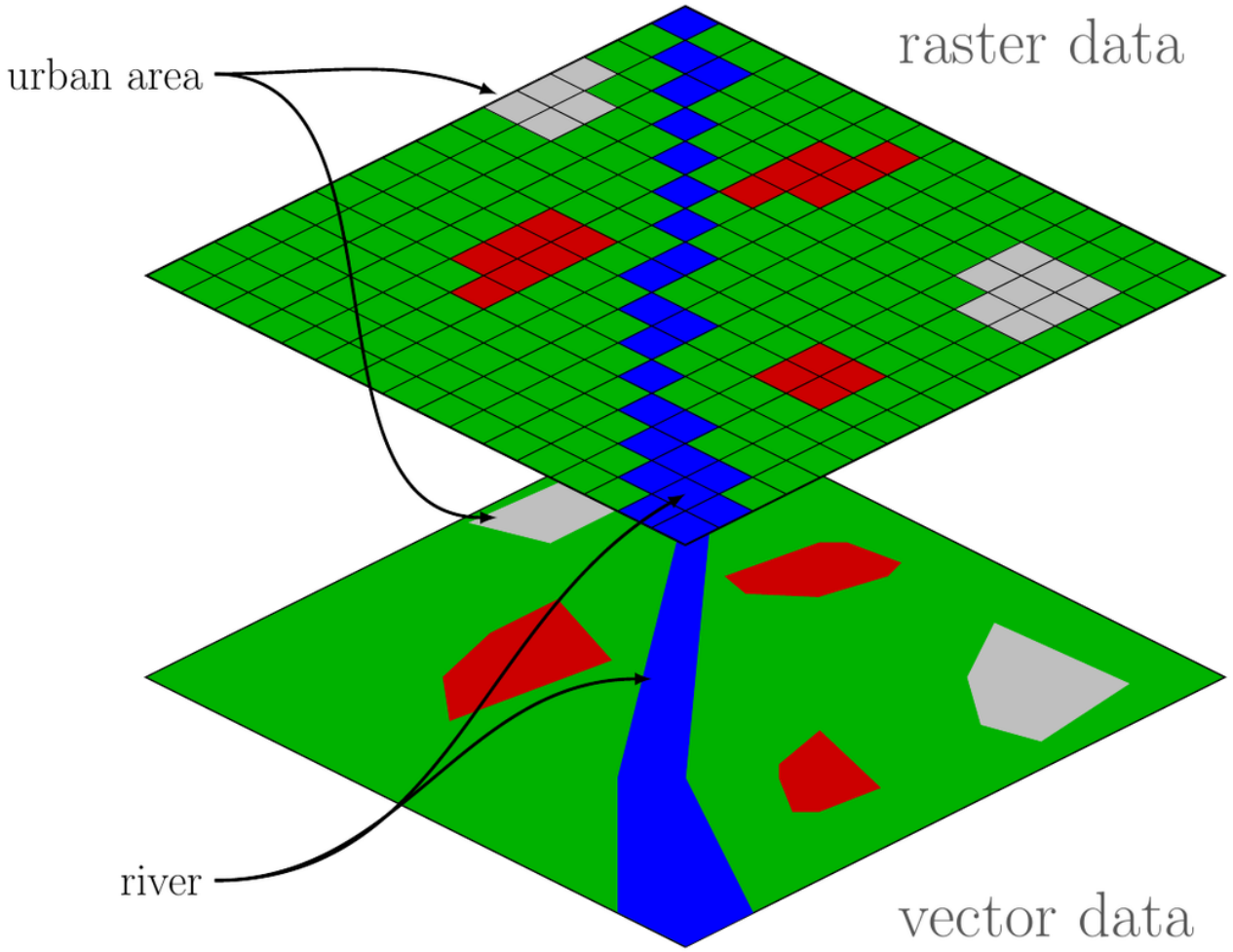
GIS data types



GIS terms

Vector data:

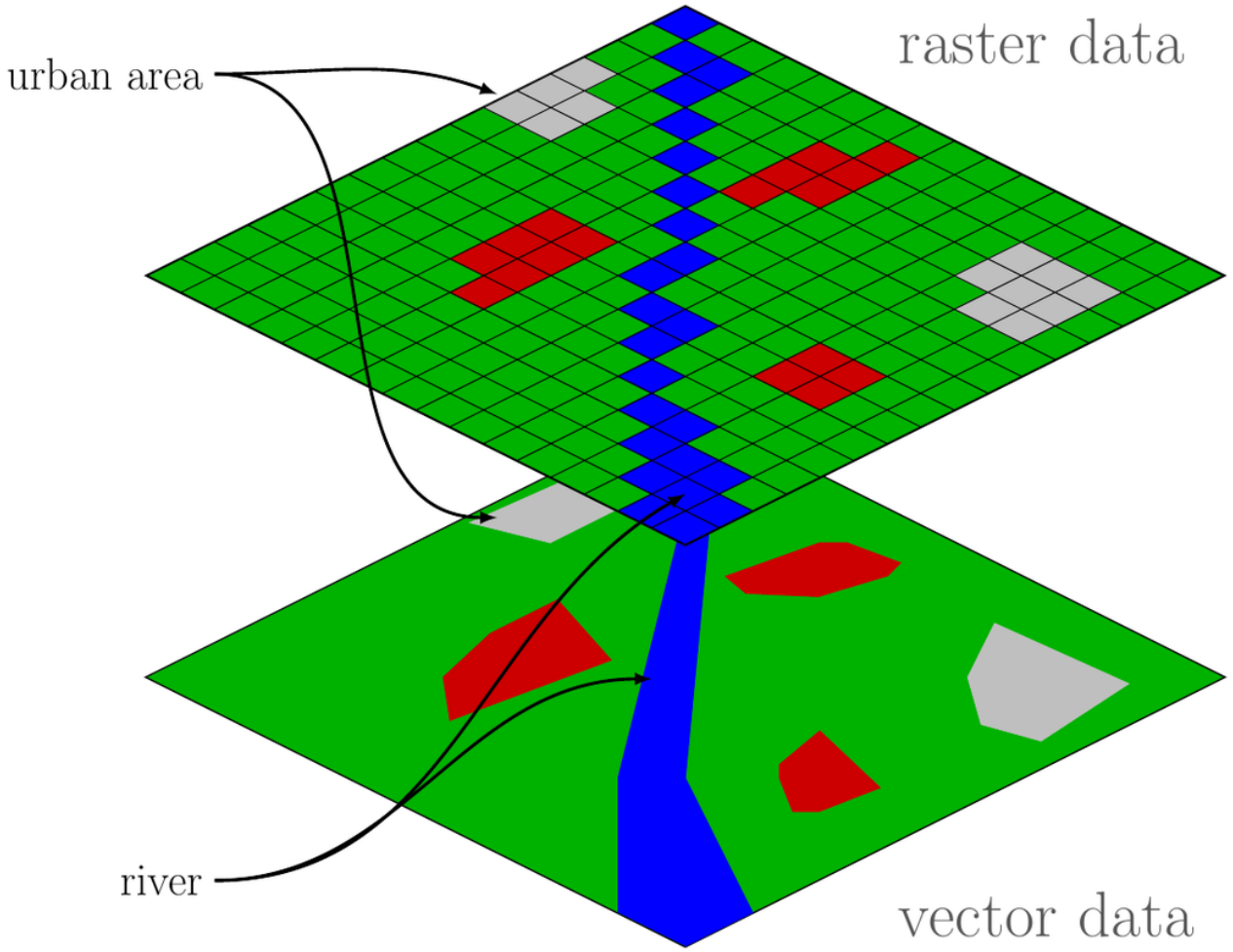
- Polygons
- Lines
- Points



GIS terms

Raster data:

- Covers large surfaces
- Note variable resolution



Natural Earth

- Natural Earth is a public domain resource for mapping
- 1:10, 1:50, and 1:110 million meter scales
  - Coarse, medium, fine
  - large, medium, small
- Includes both vector and raster data

Retrieving Natural Earth data

- ne_countries()
- ne_states()
- ne_download()

Retrieving Natural Earth data

- ne_countries()
- ne_states()
- ne_download()

Get natural earth world country polygons

Description

returns world country polygons at a specified scale, or points of tiny_countries

Usage

```
ne_countries(scale = 110, type = "countries", continent = NULL,
  country = NULL, geounit = NULL, sovereignty = NULL,
  returnclass = c("sp", "sf"))
```

Arguments

`scale`	scale of map to return, one of `110`, `50`, `10` or `'small'`, `'medium'`, `'large'`
`type`	country type, one of 'countries', 'map_units', 'sovereignty', 'tiny_countries'
`continent`	a character vector of continent names to get countries from.
`country`	a character vector of country names.
`geounit`	a character vector of geounit names.
`sovereignty`	a character vector of sovereignty names.
`returnclass`	'sp' default or 'sf' for Simple Features

Retrieving Natural Earth data

- ne_countries()
- ne_states()
- ne_download()

Get natural earth world state (admin level 1) polygons

Description

returns state polygons (administrative level 1) for specified countries

Usage

```
ne_states(country = NULL, geounit = NULL, iso_a2 = NULL, spdf = NULL,
  returnclass = c("sp", "sf"))
```

Arguments

`country`	a character vector of country names.
`geounit`	a character vector of geounit names.
`iso_a2`	a character vector of iso_a2 country codes
`spdf`	an optional alternative states map
`returnclass`	'sp' default or 'sf' for Simple Features

Retrieving Natural Earth data

- ne_countries()
- ne_states()
- ne_download()

Usage

```
ne_download(scale = 110, type = "countries", category = c("cultural",
  "physical", "raster"), destdir = tempdir(), load = TRUE,
  returnclass = c("sp", "sf"))
```

Arguments

`scale`	scale of map to return, one of `110`, `50`, `10` or `'small'`, `'medium'`, `'large'`
`type`	type of natural earth file to download one of 'countries', 'map_units', 'map_subunits', 'sovereignty', 'states' OR the portion of any natural earth vector url after the scale and before the . e.g. for 'ne_50m_urban_areas.zip' this would be 'urban_areas'. See Details. OR the raster filename e.g. for 'MSR_50M.zip' this would be 'MSR_50M'
`category`	one of natural earth categories : 'cultural', 'physical', 'raster'
`destdir`	where to save files, defaults to `tempdir()`, `getwd()` is also possible.
`load`	TRUE/FALSE whether to load file into R and return
`returnclass`	'sp' default or 'sf' for Simple Features

	scale = 'small'	scale = 'medium'	scale = 'large'
category = 'physical', type = '[below]'			
coastline	y	y	y
land	y	y	y
ocean	y	y	y
rivers_lake_centerlines	y	y	y
lakes	y	y	y
glaciated_areas	y	y	y
antarctic_ice_shelves_polys	-	y	y
geographic_lines	y	y	y
graticules_1	y	y	y
graticules_30	y	y	y
wgs84_bounding_box	y	y	y
playas	-	y	y
minor_islands	-	-	y
reefs	-	-	y
category = 'cultural', type = '[below]'			
populated_places	y	y	y
boundary_lines_land	y	y	y
breakaway_disputed_areas	-	y	y
airports	-	y	y
ports	-	y	y
urban_areas	-	y	y
roads	-	-	y
railroads	-	-	y

Working with sf objects

1. Data frame modified to contain spatial data
2. Filter variables using tidyverse approaches
3. Add variables, but use join_* to make sure they line up with existing data

Making basic maps with sf objects

1. Download/specify the relevant data
2. Pipe into ggplot
3. Call geom_sf()
4. theme_bw() or theme_void() are generally best

Useful operations with sf objects

1. Zooming in: coords_sf(ylim = c(), xlim = c())
   Note: longitude in North America is negative. 89W = -89
2. Highlighting specific areas: second geom_sf(data, fill)
3. Annotate using text, boxes, and lines with annotate()

Add point data

Latitude and longitude require a data type conversion

st_as_sf()

Example: coordDat <- st_as_sf(DF, coords = c("long", "lat"), crs = 4326)

- Input data frame
- Which variables represent longitude and latitude
- crs specifies the projection. 4326 is most universally appropriate.

Labeling

library(ggrepel)

geom_text_repel()

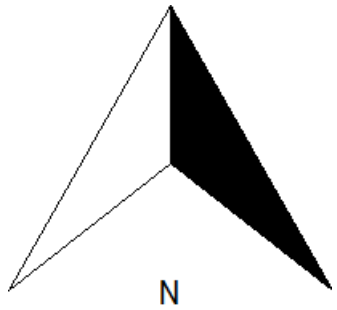
geom_label_repel()

e.g., geom_text_repel(data = Dat, aes(x = long, y = lat, label = label)

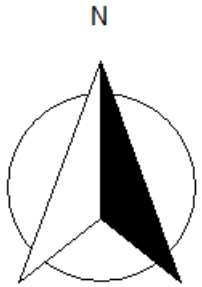
Adding a scale and compass

```
library(ggspatial)
annotation_scale()
annotation_north_arrow(location, pad_x, pad_y)
```

- location = c(“tl”, “tr”, “bl”, “br”)

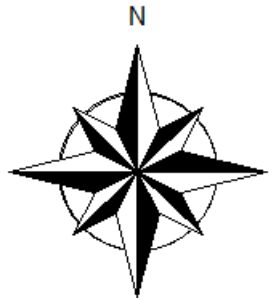


north_arrow_orienteering



north_arrow_fancy_orienteering

north_arrow_minimal



north_arrow_nautical