Spatial Data in Economics

Prof. Tzu-Ting Yang
楊子霆

Institute of Economics, Academia Sinica
中央研究院經濟研究所

March 26, 2026

Overview

Why GIS for Economists?

- The use of geographic information system (GIS) has increasingly been popular among economists
- There are at least three reasons for this popularity

1. GIS allows economists to observe the previously unobserved and makes research feasible
   - Satellite images
   - Scanned old maps
2. GIS helps economists identify causal impacts in a more credible way than previously possible
   - Distance or Elevation
   - Geographic boundaries
3. GIS enables economists to combine large-scale administrative or commercial data with spatial information
   - Mobile phone data, credit card transactions, GPS tracking

Make Research Feasible

Example: Satellite Images

J. Vernon Henderson, Adam Storeygard, and David N. Weil (2012), "**Measuring Economic Growth from Outer Space**", AER

- They develop a statistical framework to use satellite data on night lights to estimate GDP growth
  - Their estimates are quite close to official growth data
- Using night lights data, empirical analyses of growth need no longer use countries as the unit of analysis
  - Measure growth for sub- and supranational regions

Satellite Images

World

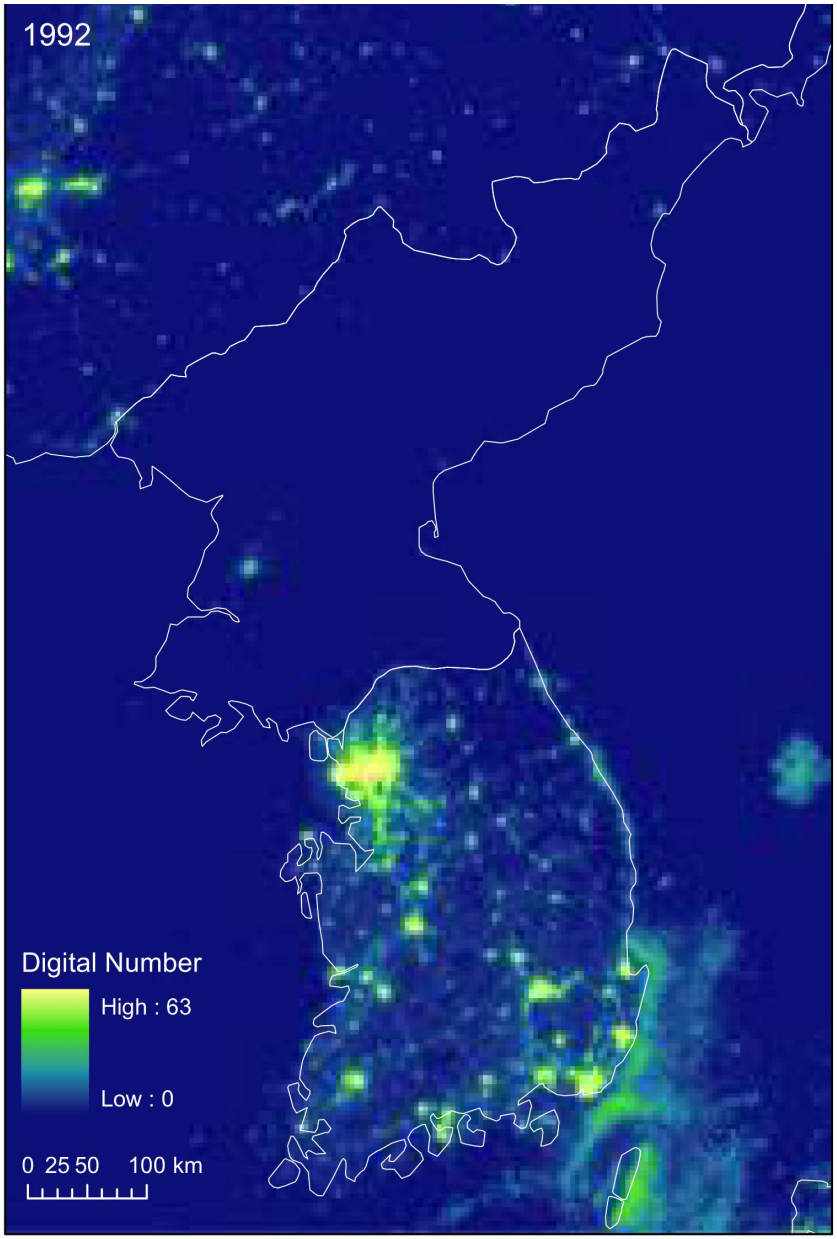
Robinson projection

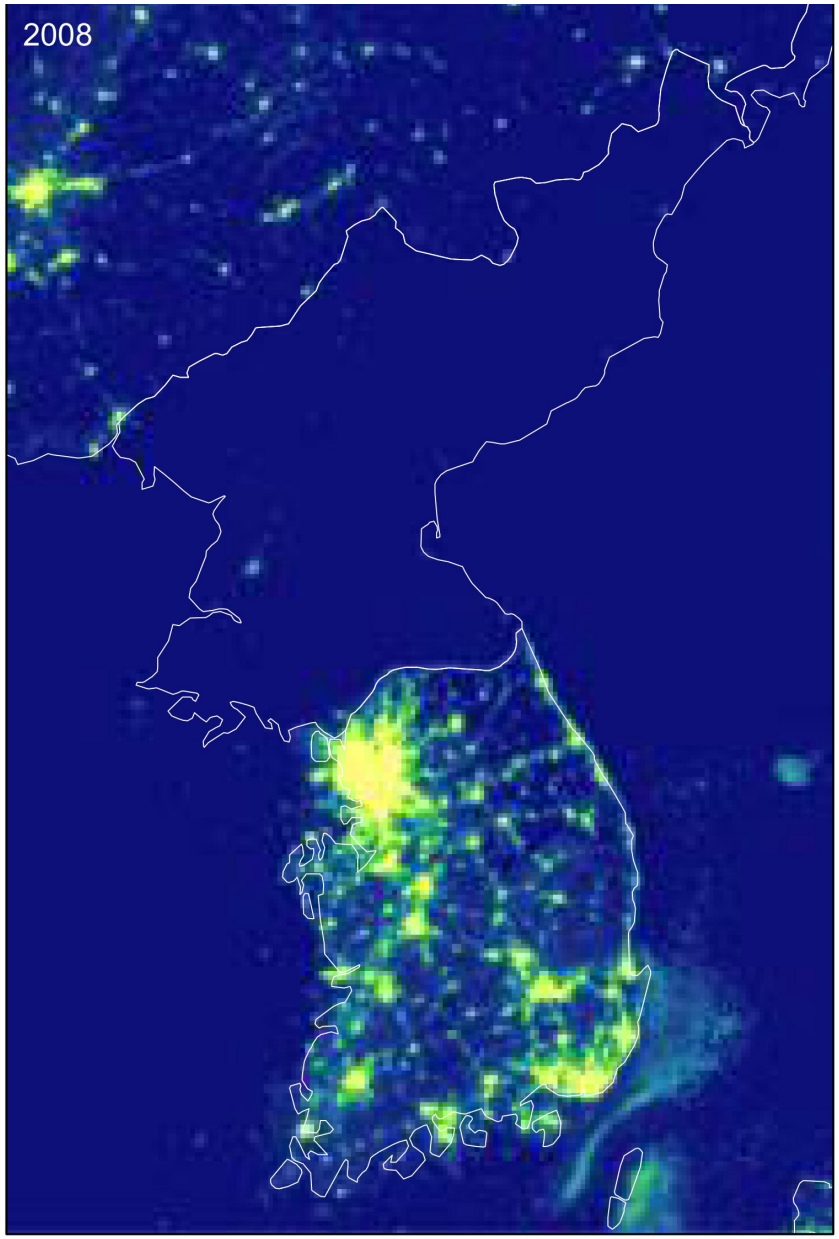
FIGURE 1. LIGHTS AT NIGHT, 2008

Source: Image and data processing by NOAA's National Geophysical Data Center. DMSP data collected by the United States Air Force Weather Agency.

Satellite Images

Long-Run Growth in Korea





Universal Transverse Mercator projection

Satellite Images

More Applications of Night Lights

- Night lights data have been applied in many contexts beyond GDP measurement
  - **Conflict and violence**: measure economic activity in war zones where official statistics are unavailable
  - **Poverty mapping**: identify subnational poverty hotspots in developing countries
  - **Urbanization**: track city growth and spatial expansion over time
- Two main data sources for night lights:
  - **DMSP-OLS** (1992–2013): eogdata.mines.edu
  - **VIIRS DNB** (2012–present): eogdata.mines.edu/products/vnl
  - **NASA Black Marble** (VIIRS, high-res): earthdata.nasa.gov

Other Satellite Data

Vegetation Index (NDVI)

- **What it measures**: the "greenness" of land surface, reflecting vegetation health and density
  - $NDVI = \frac{NIR-Red}{NIR+Red}$, ranging from -1 to $+1$
  - Higher values $\Rightarrow$ denser, healthier vegetation
  - Negative values $\Rightarrow$ water bodies
- **Data**: NASA MODIS (250m–1km, 16-day or monthly)
- **Applications in economics**:
  - Proxy for **agricultural productivity** and crop yields
  - Proxy for **rainfall** in areas with sparse weather stations
  - Instrument for **income shocks** in conflict and migration research
- **Source**: modis.gsfc.nasa.gov

Other Satellite Data

Land Surface Temperature (LST)

- **What it measures**: temperature of the Earth's surface itself (not air temperature)
  - Derived from thermal infrared radiation emitted by the surface
  - Surface temperature can differ substantially from air temperature (e.g., asphalt on a hot day)
- **Data**: NASA MODIS MOD11 (1km, daily or 8-day composites)
- **Advantage over weather stations**: globally uniform coverage, no interpolation needed
- **Applications in economics**:
  - **Urban heat islands**: link green space and land use to housing prices
  - **Labor productivity**: estimate causal effect of heat on wages and output
  - **Energy demand**: high temperatures raise cooling demand and electricity consumption
- **Source**: modis.gsfc.nasa.gov

Other Satellite Data

Aerosol Optical Depth (AOD)

- **What it measures**: how much sunlight is blocked by suspended particles across the entire atmospheric column
  - AOD $= 0$: clean, transparent atmosphere
  - AOD $= 1$: heavy particulate loading (smoke, dust, pollution)
- **Data**: NASA MODIS MOD04 (3km–10km, daily)
- **Why use AOD instead of PM2.5?**
  - Ground PM2.5 monitors are sparse, especially in developing countries
  - AOD provides global, uniform coverage since 2000
  - Caveat: AOD measures the full atmospheric column; conversion to ground-level PM2.5 requires additional modeling
- **Applications in economics**:
  - Pollution effects on **health, cognition, and productivity**
  - Evaluation of **environmental regulations**
- **Source**: modis.gsfc.nasa.gov · All products free at earthdata.nasa.gov

Make Research Feasible

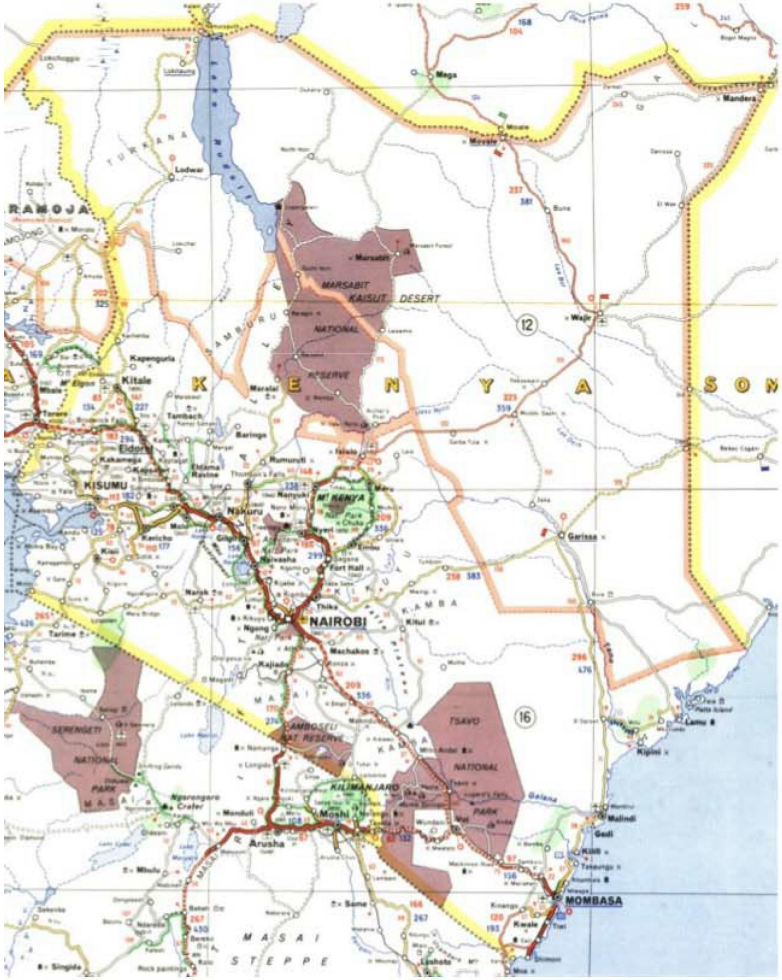
Example: Scanned Old Maps

Robin Burgess, Remi Jedwab, Edward Miguel, Ameet Morjaria, and Gerard Padró i Miquel (2015), "**The Value of Democracy: Evidence from Road Building in Kenya**", AER

- Did Kenyan presidents build more roads for their co-ethnics?
  - They digitalized the old maps to get the building of roads since the year of independence in Kenya (1961)
  - Track road network expansion over time

Scanned Old Maps

Map of Roads in Kenya in 1961



Michelin map in 1961

Digitization and Standardization in GIS

Make Research Feasible

Example: Scanned Old Maps

- They find strong evidence of ethnic favoritism
- Districts that share the ethnicity of the president:
  - Receive twice as much expenditure on roads
  - Have five times the length of paved roads built
  - This favoritism disappears during periods of democracy

New Data Sources

Beyond Maps and Satellites

- Recent advances in data collection have opened new frontiers for spatial economics
- **Mobile phone / GPS data**
  - Track individual mobility patterns, commuting flows, and migration
  - Estimate local economic activity via foot traffic
- **Geotagged social media and text data**
  - Measure consumer sentiment, disaster response, or social interactions at the neighborhood level
- **Historical aerial photographs and colonial records**
  - Reconstruct pre-independence land use, property rights, and infrastructure
  - Similar to the Kenya road map digitization approach
- These data sources are often combined with GIS to link spatial and non-spatial information

Identify Causal Impacts

Why Geography Helps

- A central challenge in economics: policies, treatments, and exposures are **not randomly assigned**
- GIS provides tools to construct **credible comparison groups** using geographic variation:
  - **Distance** as an instrumental variable
    - ★ Distance to a trade hub, river, coast, or infrastructure node affects treatment but is plausibly exogenous
  - **Elevation / terrain** as an instrumental variable
    - ★ Ruggedness or slope affects infrastructure rollout costs
  - **Administrative boundaries** for Spatial RDD
    - ★ Policies often change sharply at borders
    - ★ Locations just inside vs. just outside a border are similar in all other respects

Identify Causal Impacts

Example: Distance

Nathan Nunn (2008), "**The Long-term Effects of Africa's Slave Trades**", QJE

- Can part of Africa's current underdevelopment be explained by its slave trades ?
  - More slave trades leads to lower growth in Africa
  - Since slave trades may be endogenous to the historical level of development
- IV for slave trade:
  - Slave trade: the number of exported slaves from each country.
  - IV: distance to the nearest slave trade center in the Americas

Use Distance as an IV

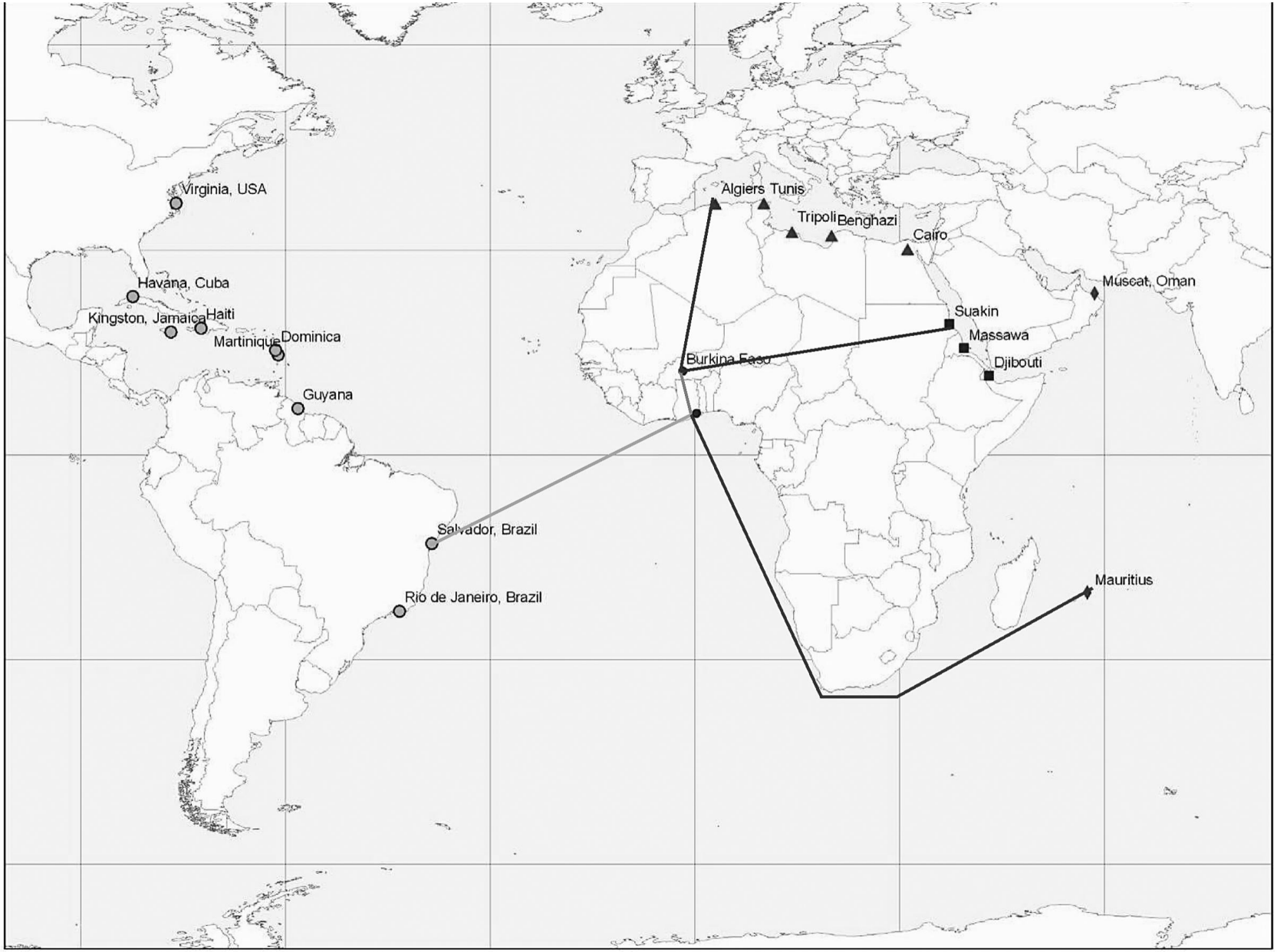


FIGURE V

Identify Causal Impacts

Example: Distance

- Location of slave trade centers:
  - Determined by climate suitability of plantation crops/location of mines
  - Not affected by the distance to Africa
  - Distance to slave markets $\neq$ Distance to other economic opportunities
- The author finds that countries with more slaves exported are poorer today

Identify Causal Impacts

Example: Land Slope

Taryn Dinkelman (2011), "**The Effects of Rural Electrification on Employment: New Evidence from South Africa**", AER

- This paper estimates the impact of electrification on female labor supply
- IV for electrification: mean land slope
  - Flat terrain $\Rightarrow$ cheaper to lay power lines $\Rightarrow$ more likely to be electrified
  - Land slope is determined by geology, not by economic conditions
- They find that electrification significantly raises female employment by:
  - Releasing women from home production
  - Enabling microenterprises

Use Elevation as an IV

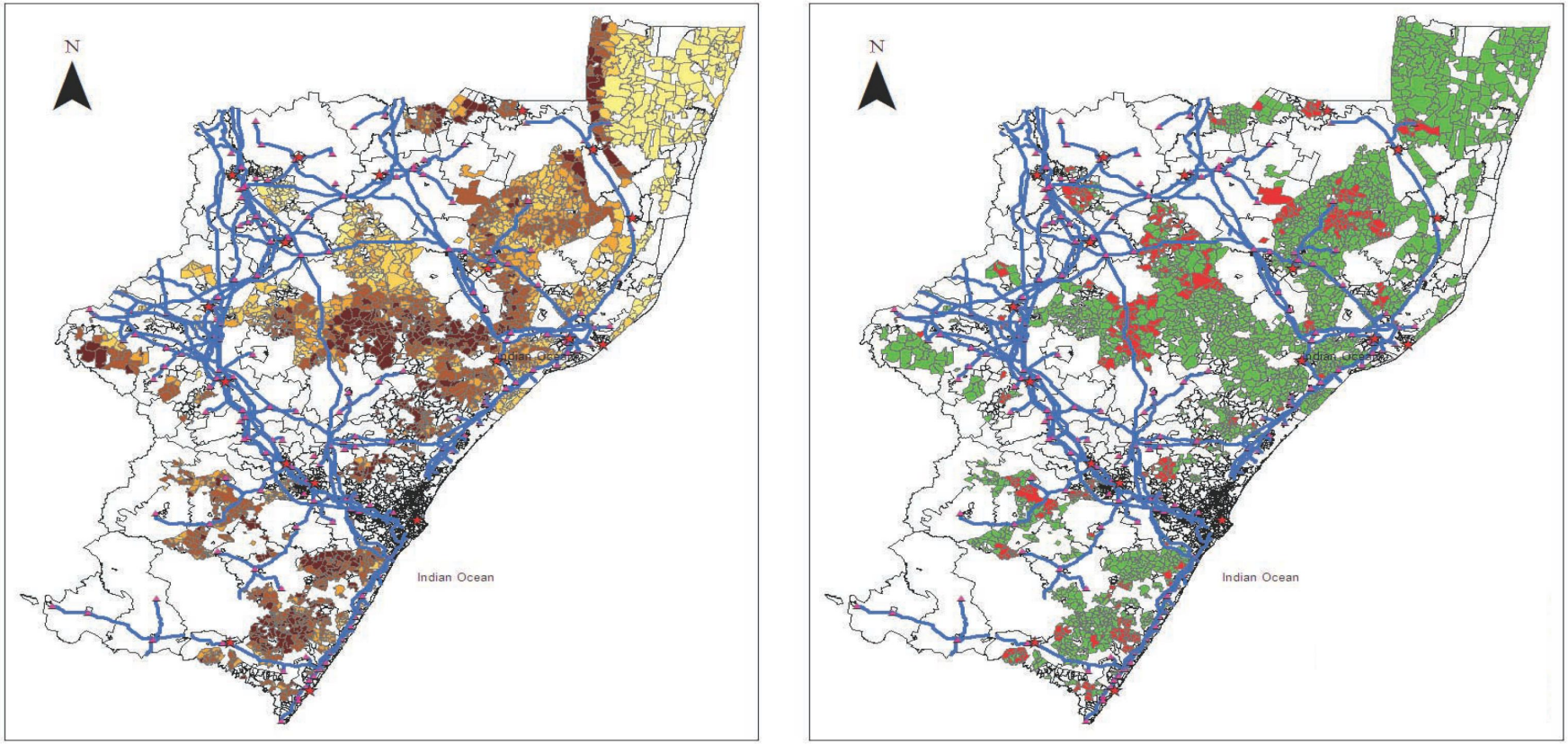


FIGURE 2. SPATIAL DISTRIBUTION OF GRADIENT AND ELECTRICITY PROJECT AREAS IN KWAZULU-NATAL

Notes: Shaded communities are in the analysis sample ($N = 1{,}816$). Thick lines depict electricity gridlines in 1996, triangles are electricity substations in 1996, and stars represent towns. Gradient is shown in the figure on the left: steeper areas are shaded dark, flatter areas are shaded light. Electricity Project areas are depicted in the figure on the right: project areas are shaded dark, lighter shaded areas are electrified after 2001 or not at all.

- Slope (lighter = flatter)

Rafael Lalive (2008), "**How do extended benefits affect unemployment duration? A regression discontinuity approach**", Journal of Econometrics

- This paper examine the effect of extended benefits on unemployment duration using spatial RDD
  - Sharp discontinuities in treatment assignment at age 50 and at the border between eligible regions and control regions
  - Extend the maximum duration of unemployment benefits from 30 weeks to 209 weeks

Spatial Regression Discontinuity Design

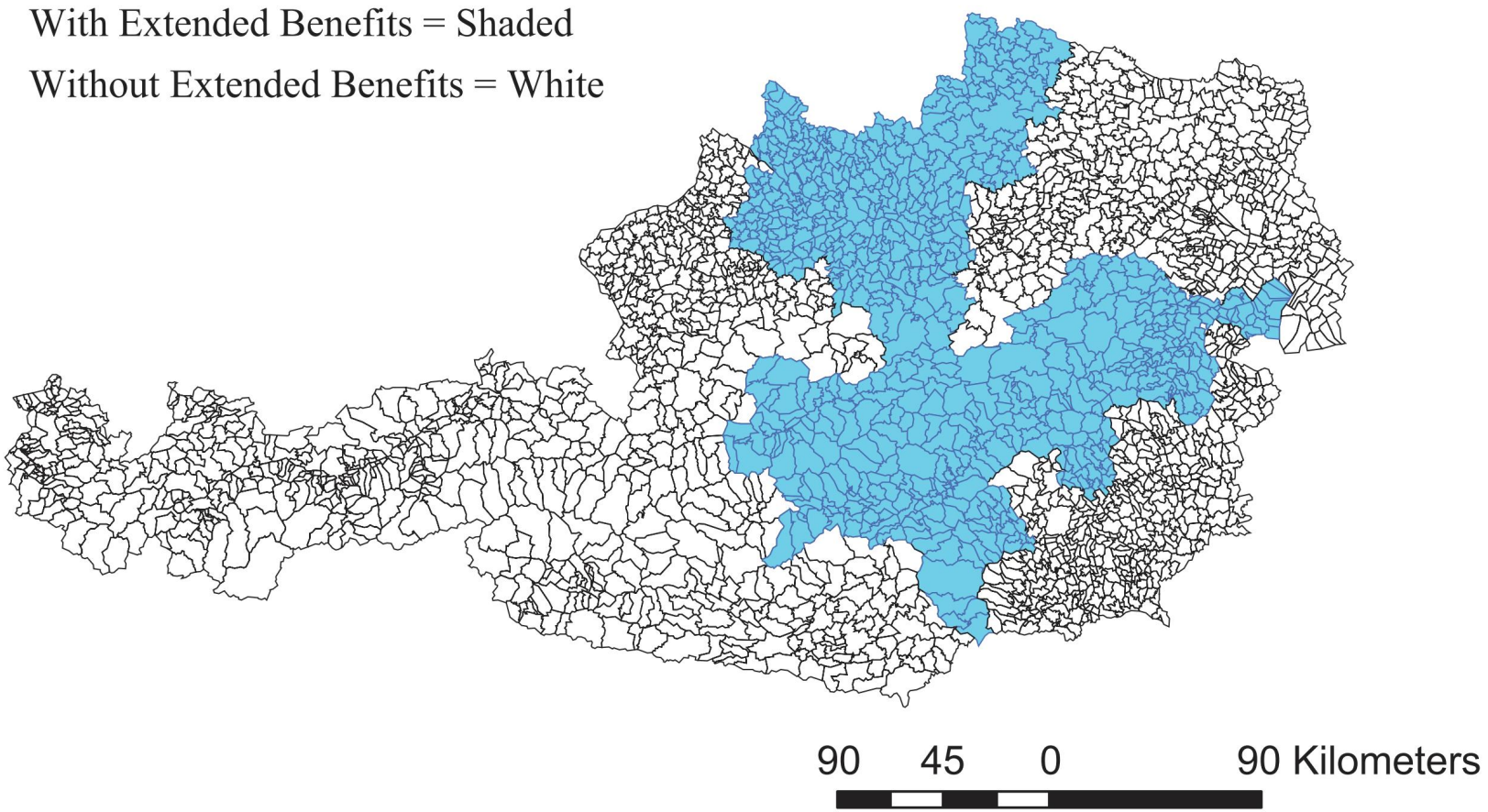
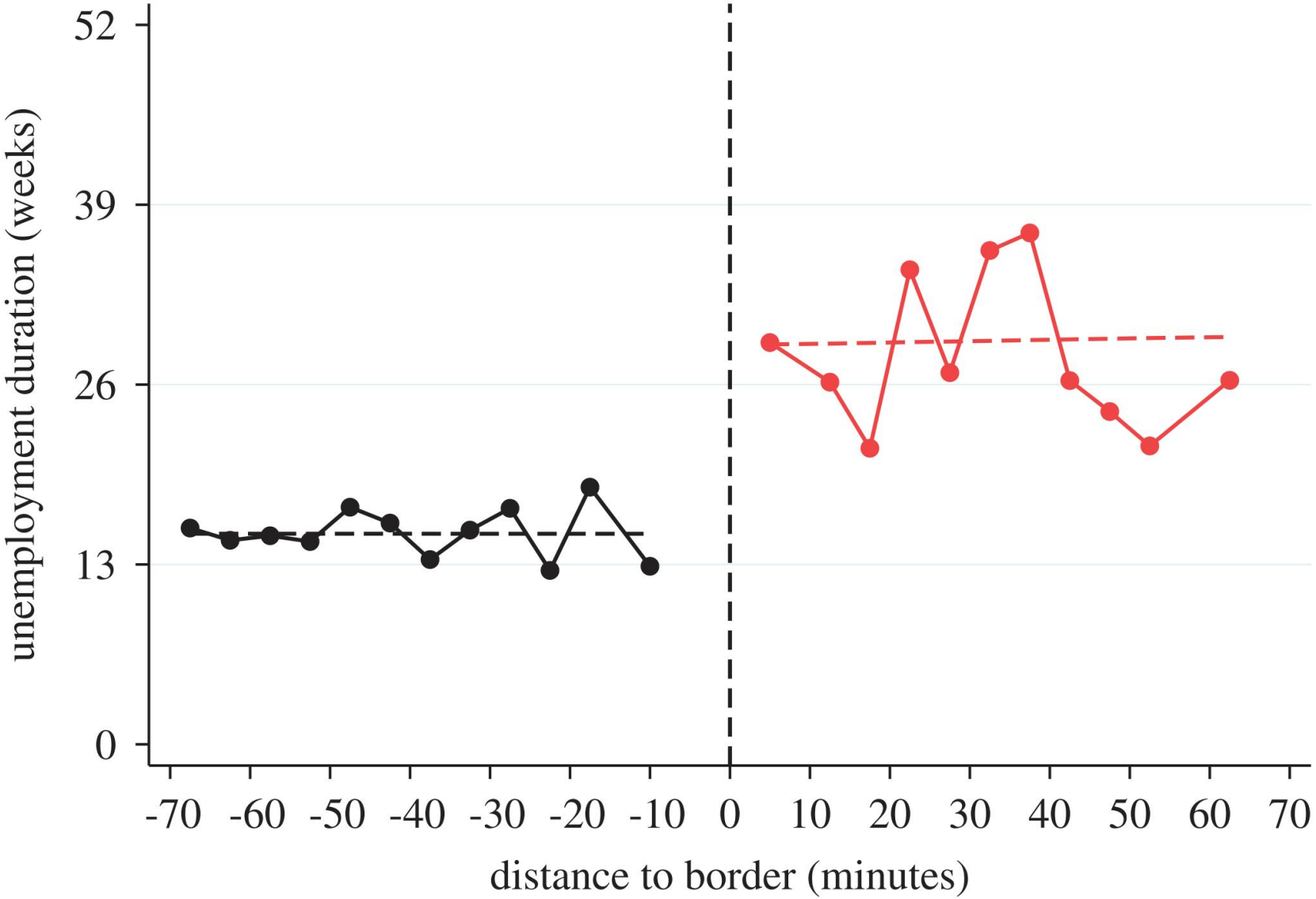


Fig. 1. Regional distribution of REBP.

Spatial Regression Discontinuity Design



Introduction to Spatial Data

Introduction to Spatial Data

Overview

- Spatial data usually focus on **locations**
- Spatial data (phenomenon) generally have two primary types:
  1. Discrete phenomenon
     - ★ It is also called **spatial objects**
     - ★ It has boundaries
     - ★ Examples: river, road, country or town
     - ★ Usually use **vector data**
  2. Continuous phenomenon
     - ★ It is also called **spatial fields**
     - ★ It does not have natural boundaries
     - ★ Examples: light intensity at night, elevation, temperature, or air quality (PM 2.5)
     - ★ Usually use **raster data**

Vector Data

Spatial Objects

- **Spatial objects** are usually represented by **vector data**
- Information about **spatial objects** can be classified into two categories:
  - Spatial attributes
    - ★ Coordinates that represent its location
    - ★ Coordinates that represent its shape
  - Non-spatial attributes
    - ★ Township name
    - ★ Population size
    - ★ GDP per capita

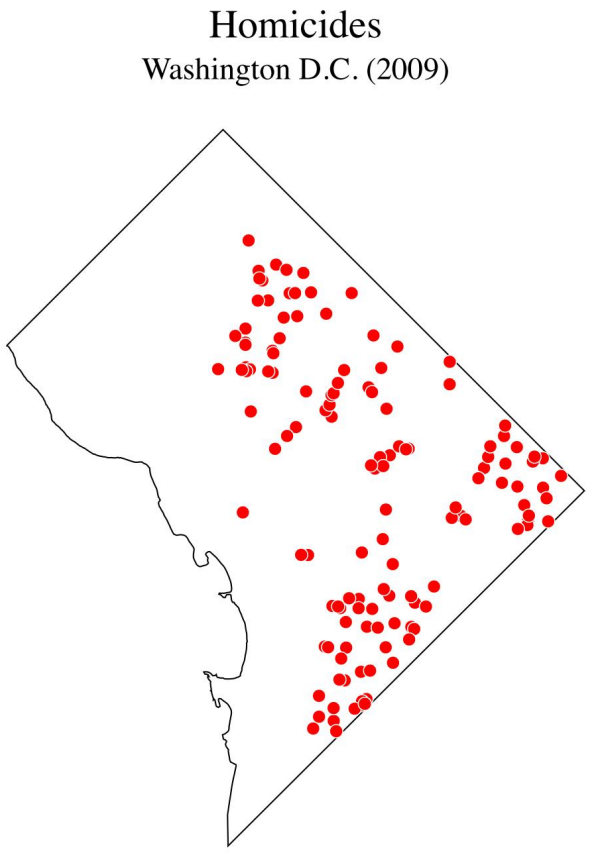
Types of Vector Data

- According to their spatial attributes, **spatial objects** can be classified into several types
- Here, we focus on two basic types:
  - Points (point data)
  - Polygons (area data)
- In all cases, the geometry of these data structures consists of sets of **coordinate pairs (x, y)**

Vector Data

Points

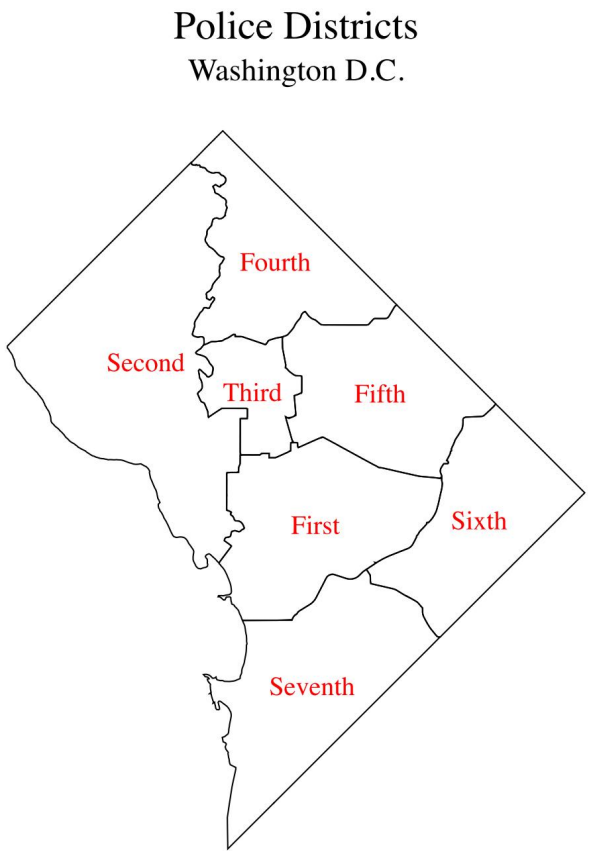
- A point s_i is a spatial object located within study area $\mathcal{A}$ at coordinates (s_{i1}, s_{i2})
- Points can represent the following real entities:
  - Buildings
  - Places where specific events took place
  - Pollution sources
- Each point might also have **non-spatial attributes**, such as household size or number of people.



Vector Data

Polygons

- A polygon r_i is a *region* of study area $\mathcal{A}$ bounded by a closed polygonal chain
- It can be defined by the coordinate set $\{(r_{i1(1)}, r_{i2(1)}), (r_{i1(2)}, r_{i2(2)}), ..., (r_{i1(m)}, r_{i2(m)}), ..., (r_{i1(M)}, r_{i2(M)})\}$, where $r_{i1(1)} = r_{i1(M)}$ and $r_{i2(1)} = r_{i2(M)}$
- Polygons can represent several kinds of real entities:
  - States, counties, electoral districts
  - Parks, lakes

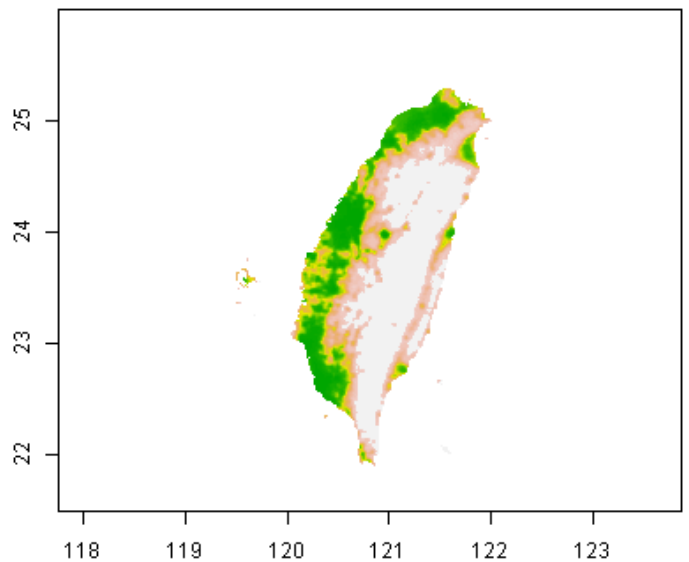


- **Spatial fields** are usually represented by **raster data**
- Raster data is commonly used to represent continuous variables
  - A raster divides the world into a grid of equally sized rectangles (i.e. grid cells or pixels)
  - These rectangles all have a values (or a missing value) for the variables of interest
  - A raster cell/pixel value should normally represent the value for the area it covers
    - ★ **Average**: for continuous variables (e.g., temperature, elevation)
    - ★ **Majority**: for categorical variables (e.g., land use type)

Raster Data

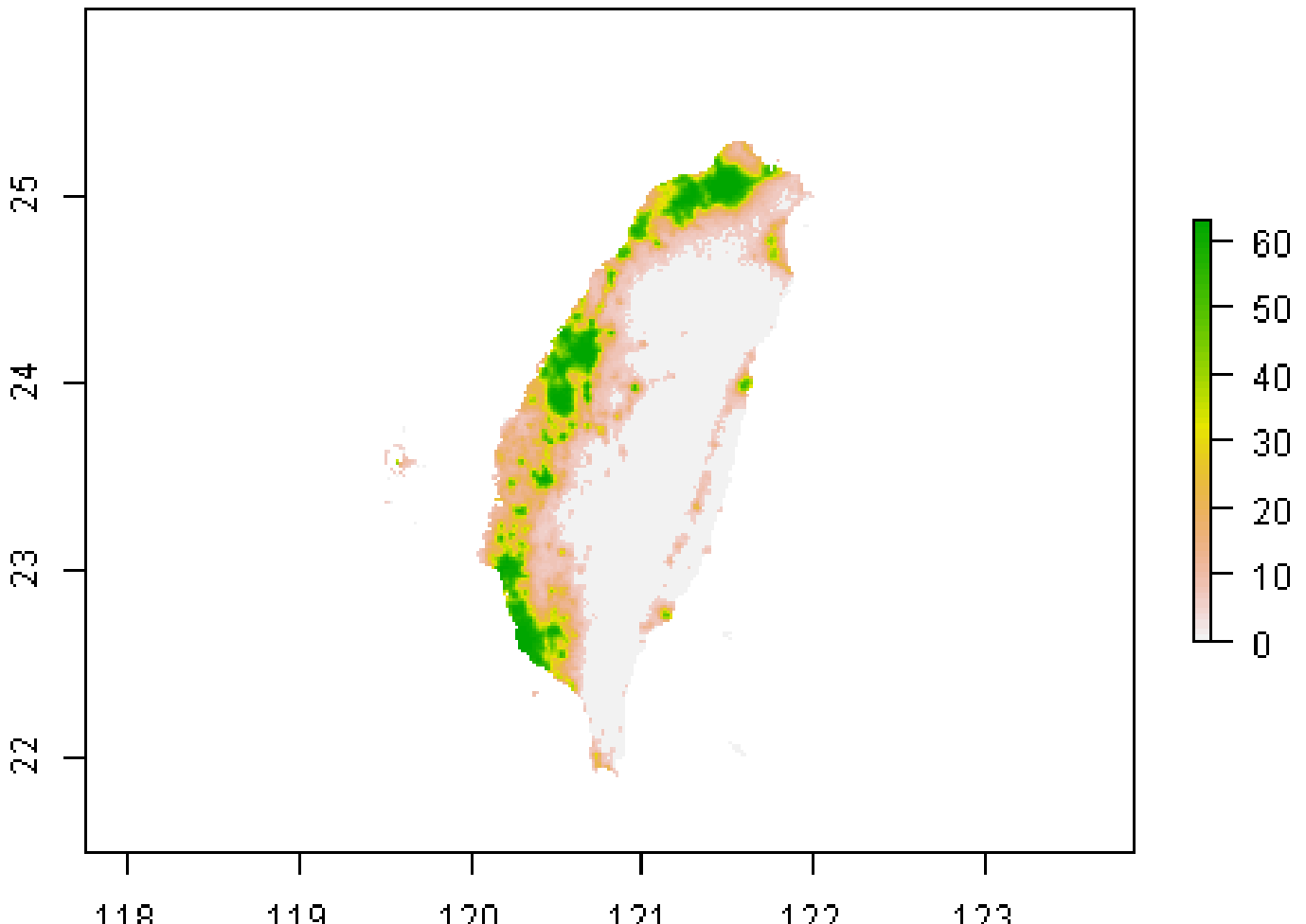
Night Light from Satellite Images

- Each raster pixel value in a **DMSP/OLS** satellite image ranges from 0 to 63 (Digital Number)
- It represents the light intensity at night in Taiwan



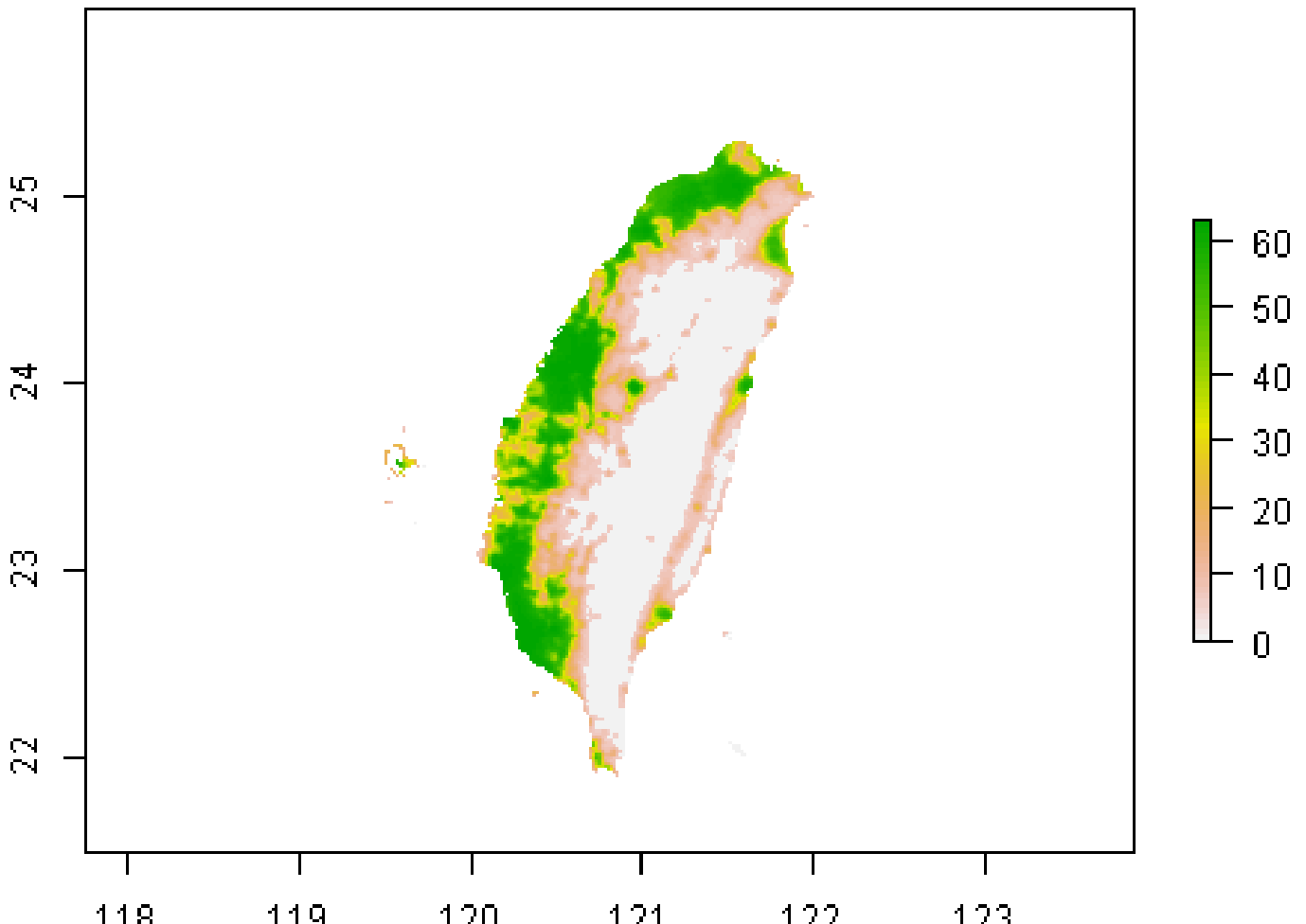
Night Light from Satellite Images

Taiwan in 1992



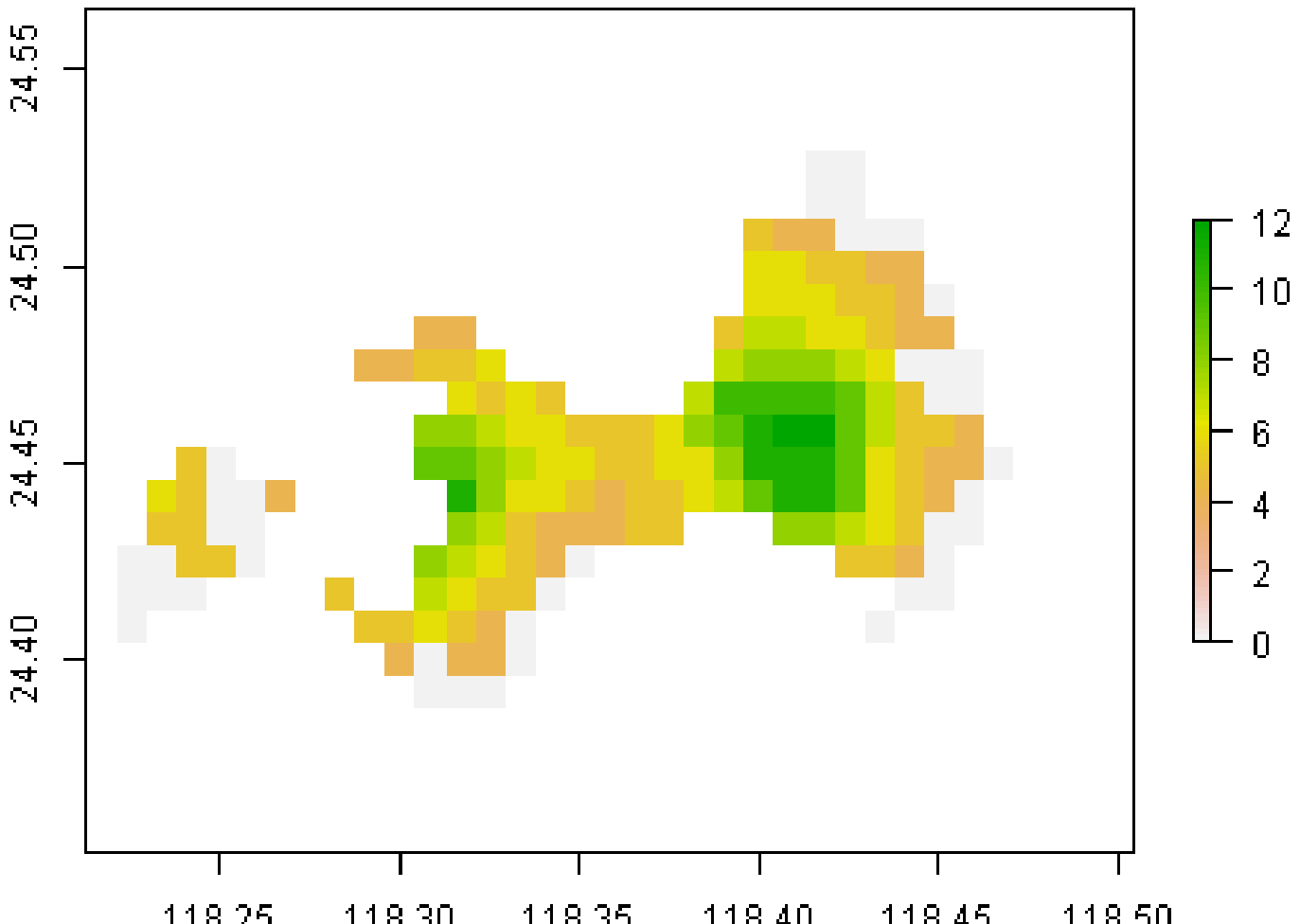
Night Light from Satellite Images

Taiwan in 2013



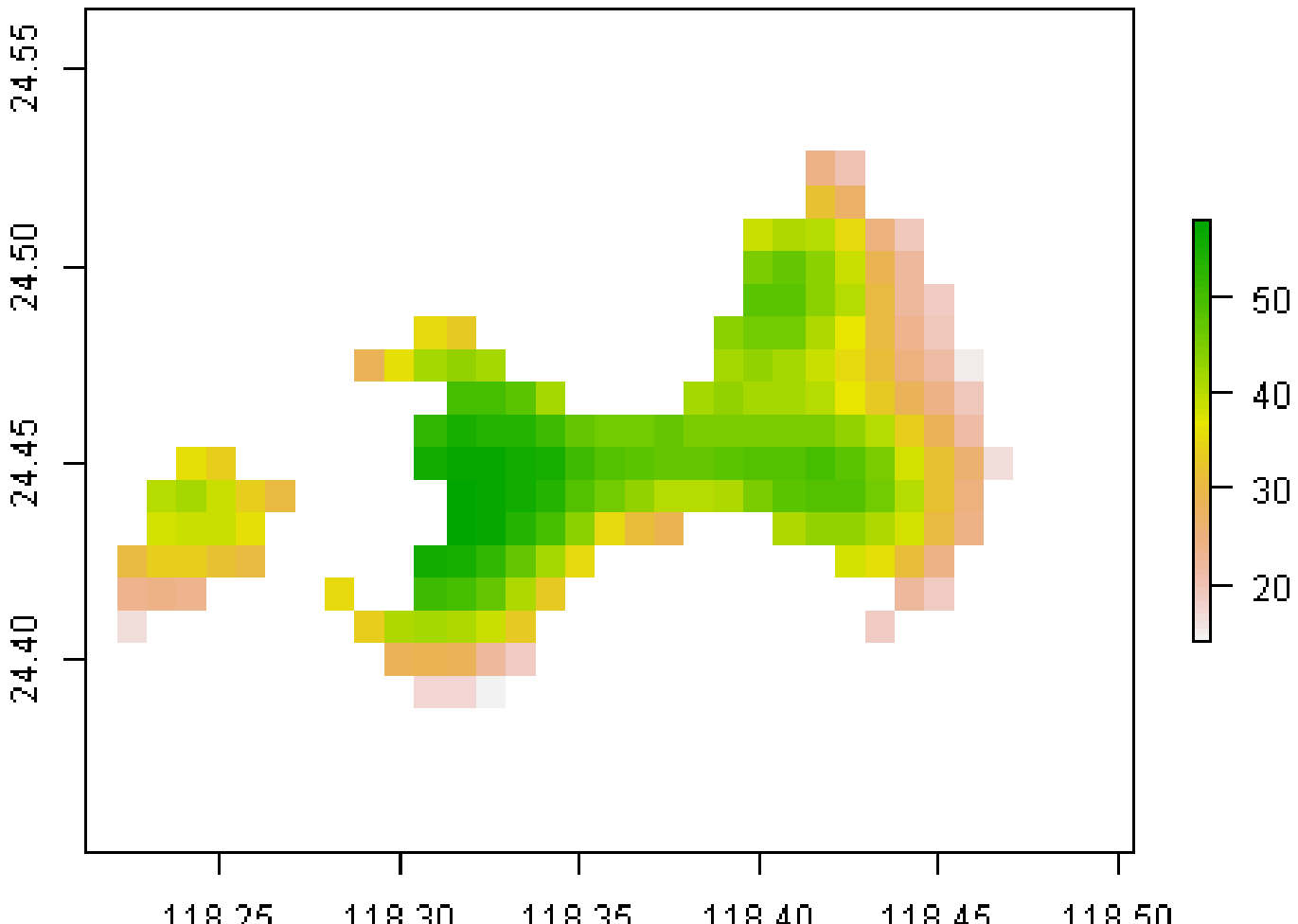
Night Light from Satellite Images

Kinmen in 1992



Night Light from Satellite Images

Kinmen in 2013



Spatial Data and Coordinate Systems

Coordinate Systems

- The distinguishing feature of spatial data is that each data point has a **geographic identifier** attached to it
- To view and manipulate your data, these identifiers will need to be referenced to a **coordinate system**
- Using the wrong coordinate system is a critical mistake that can result in your calculations being complete garbage

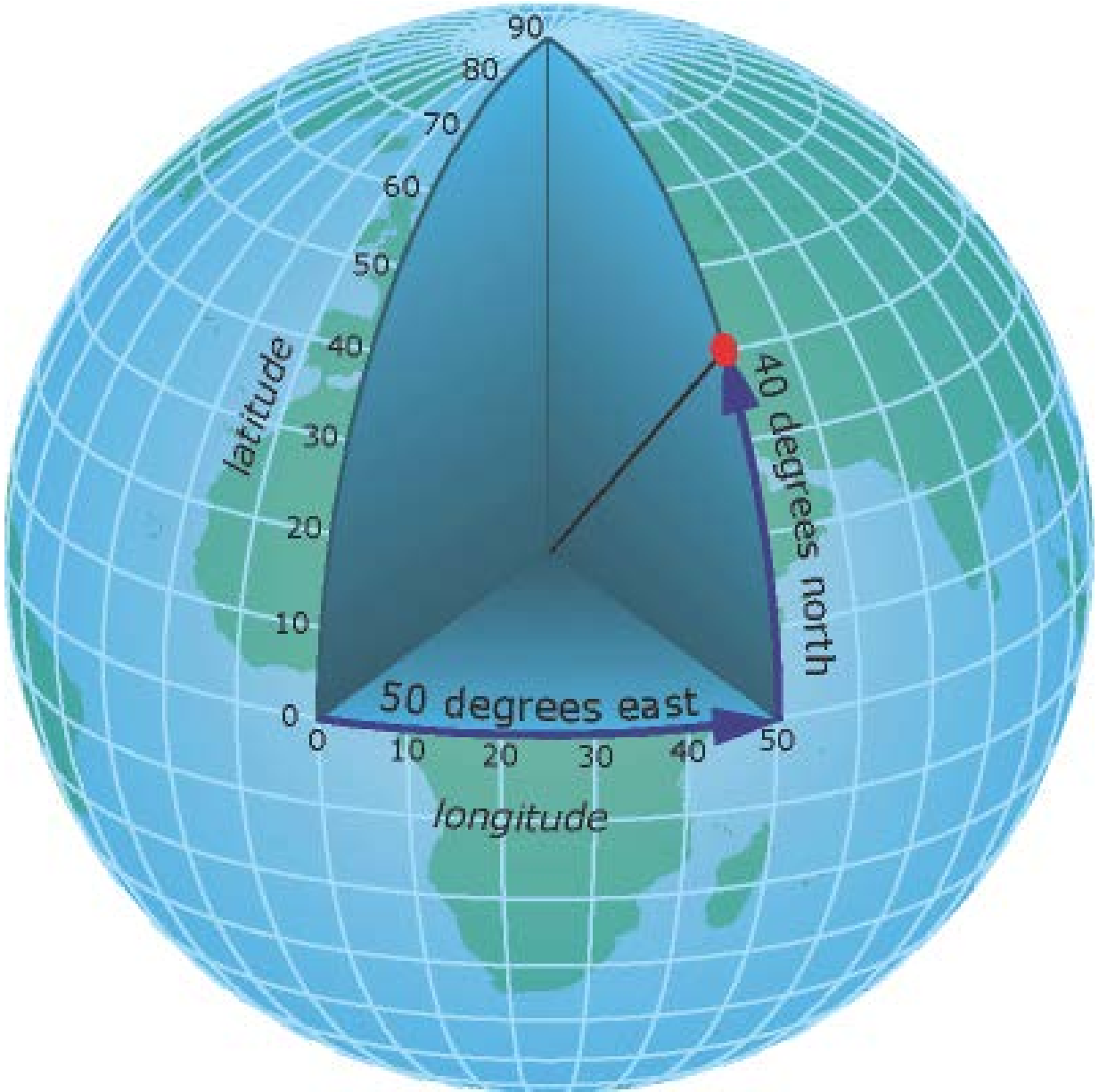
Coordinate Systems

- Two types of coordinate systems:
  1. Geographic coordinate systems
     - ★ Three dimensional angular system
  2. Projected coordinate systems
     - ★ Two dimensional planar system

Geographic Coordinate Systems

- In geographic coordinate systems (GCS), each location is coded by **angle** from **earth center**
- The unit of measure in geographic coordinate systems is **decimal degrees**
  - Taipei: 24.5723^o N (latitude) and 121.3213^o E (longitude)
  - Stockholm: 59.3293^o N and 18.0686^o E
- As with latitude and longitude, the values are bounded by -90^o to 90^o and -180^o to 180^o respectively.
- Latitude and longitude are usually expressed in that sequence, latitude before longitude

Geographic Coordinate Systems



Source: ArcGIS for Desktop

Geographic Coordinate Systems

- Latitudes:
  - North of the equator: Positive
  - South of the equator: Negative
- Longitudes:
  - East of Prime Meridian: Positive
  - West of Prime Meridian: Negative

Geographic Coordinate Systems

- Most popular GCS:
  - World Geodetic System (**WGS 1984**)
  - WGS 1984 is a coordinate system designed by the U.S. military
- GCS used in Taiwan:
  - **TWD97**: current standard adopted in 1997; closely aligned with WGS 1984 — use this for new projects
  - **TWD67**: older standard; coordinates can differ from TWD97 by ~200–800 meters — mixing the two causes errors

Geographic Coordinate Systems

- Although longitude and latitude can locate exact positions on the surface of the globe, they are NOT uniform units of measure
  - The circumference of the Earth at the equator is 40,075,161.2 meters
  - The equator is divided into 360 degrees of longitude
  - So each degree at the equator represents 111,319.9 meters or approximately 111.32 km
  - One degree of **latitude** is also $\approx$ 111 km everywhere — it is nearly constant
  - One degree of **longitude** varies: $\approx$ 111 km at the equator, but **shrinks toward zero** at the poles

Geographic Coordinate Systems

- Only along the equator does the distance represented by **one degree of longitude** approximate the distance represented by **one degree of latitude**
- As the meridians converge toward the poles, the distance represented by **one degree of longitude decreases to zero**
- A major question in spatial analysis is how to transform this **three dimensional angular system** to a **two dimensional planar system**

Projected Coordinate Systems

- **Projected coordinate systems** project the **round surface** of the earth (in degrees) onto a **flat surface** (two-dimensional)
- A **projected coordinate system** is always based on a **geographic coordinate system**
- Unlike a geographic coordinate system, a projected coordinate system has constant lengths, angles, and areas across the two dimensions
- The unit of measure in a projected coordinate system is the **meter**

Projected Coordinate Systems

- Each position has two values (x,y coordinates) that reference it to that central location
  - x specifies its horizontal position and y specifies its vertical position
  - The two values are called the x-coordinate and y-coordinate.
  - The coordinates at the origin are $x = 0$ and $y = 0$
- All projections introduce some distortions — think of peeling an orange and trying to lay the skin flat
- Different projections preserve different properties:
  - **Shape** (conformal): angles are correct, e.g., Google Maps
  - **Area** (equal-area): useful for comparing region sizes
  - **Distance** (equidistant): distances from a reference point are correct
  - **Direction** (azimuthal): bearings from center are correct

Projected Coordinate Systems

Common Examples

- **UTM (Universal Transverse Mercator)**
  - Divides the world into 60 zones; each zone uses meters as the unit
  - Taiwan is in **Zone 51N**
  - Widely used in global research; minimizes distortion within each zone
- **TWD97 / TM2** (Taiwan-specific)
  - Official projected coordinate system for Taiwan
  - Based on Transverse Mercator projection; unit is **meters**
  - Used in government maps, land administration, and most Taiwan GIS data
  - **Practical tip**: when working with Taiwan data, always check whether the file uses TWD97 (degrees) or TWD97/TM2 (meters)

Spatial Data Analysis and R

Why R for Spatial Analysis?

- R is a powerful tool for spatial analysis
- **Open Source:** R is freely available and has a large user community.
- **Comprehensive Packages:** R has numerous packages for spatial analysis (e.g., `sf`, `spatial`, `raster`).
- **Data Visualization:** R offers excellent data visualization capabilities for spatial data.

Why R for Spatial Analysis?

- **Integration:** R seamlessly integrates with other data analysis and statistical tools.
- **Reproducibility:** R allows for reproducible research through scripts (programming)
  - Clicking the ArcGIS (or QGIS) user interface is a waste of time from the perspective of academic productivity
- **Community Support:** Active R user community provides help and resources.
  - Chatgpt can help, too

sf Package

Load required packages

```
install.packages("pacman")
library(pacman)
p_load(
dplyr, tidyverse, sf, rdrobust, readxl, fastDummies, writexl
    , readr
)
```

- In R, we use **sf** package to handle and operate on spatial datasets
- The **sf** package uses the class of simple feature (sf) for spatial objects in R

Path

```
path <- "C:\\nest\\Dropbox\\Courses\\
    Causal_Inference_Big_Data\\Slides\\
    L10_spatial_data_analysis\\data\\"
path_mrt_line <- str_c(path, "MRT_1120412.csv")
path_house    <- str_c(path, "taipei_housing.csv")
```

Spatial Data Structure

Spatial Data

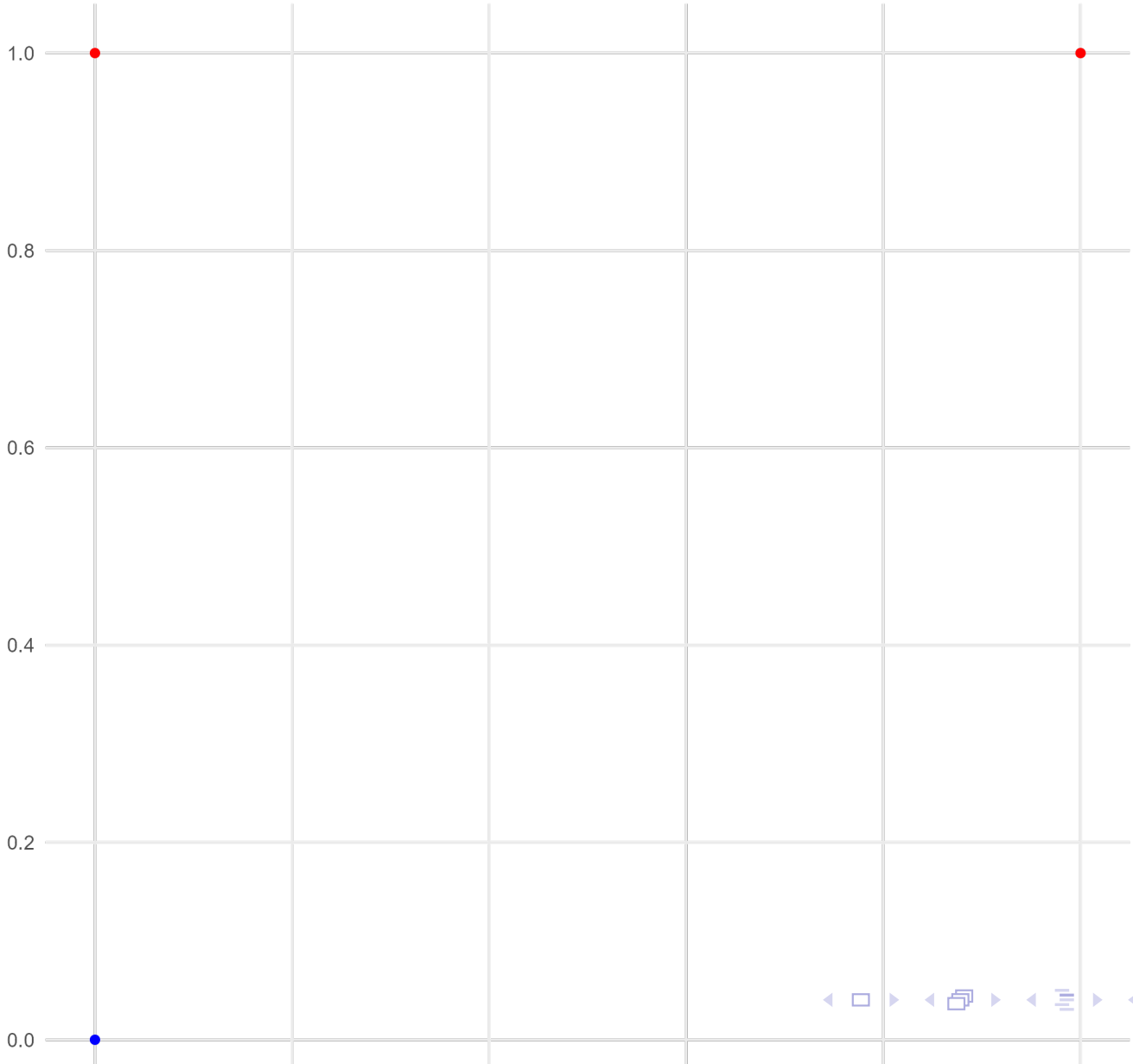
Points

```
1 one_p   <- st_point(c(0, 0))
2 multi_p <- st_multipoint(rbind(c(1, 1), c(0, 1)))
```

- **st_point** creates a single spatial point with coordinates (0, 0) and assigns it to the variable ‘one_p‘.
- **st_multipoint** creates a multipoint geometry with two points: (1, 1) and (0, 1)
  - It uses ‘**rbind**‘ to combine the coordinates into a matrix and assigns the result to the variable ‘multi_p‘.

Spatial Data

Points



Spatial Data

Lines

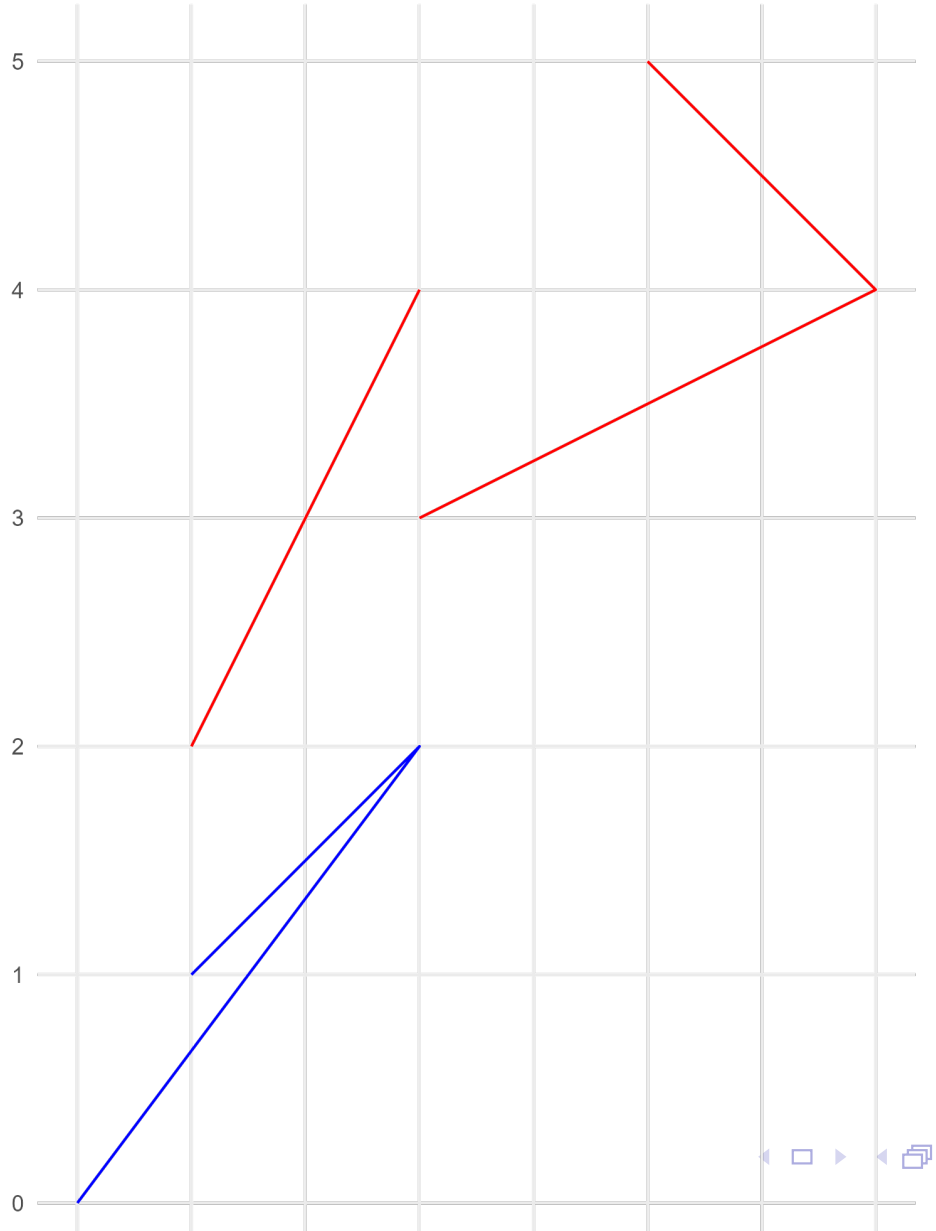
```
m1 <- rbind(c(1, 1), c(2, 2), c(0.5, 0))
m2 <- rbind(c(2, 3), c(4, 4), c(3, 5))
m3 <- rbind(c(2, 4), c(1, 2))

one_l   <- st_linestring(m1)
multi_l <- st_multilinestring(list(m2, m3))
```

- `m1` is defined as a matrix containing three coordinate pairs: (1, 1), (2, 2), and (0.5, 0).
  - `m2` and `m3` are defined in the similar way
- **st_linestring(m1)** creates a single spatial linestring object using the coordinates in `m1`.
- **st_multilinestring(list(m2, m3))** creates a multilinestring geometry by combining two linestrings defined in `m2` and `m3`.

Spatial Data

Lines



Spatial Data

Polygons

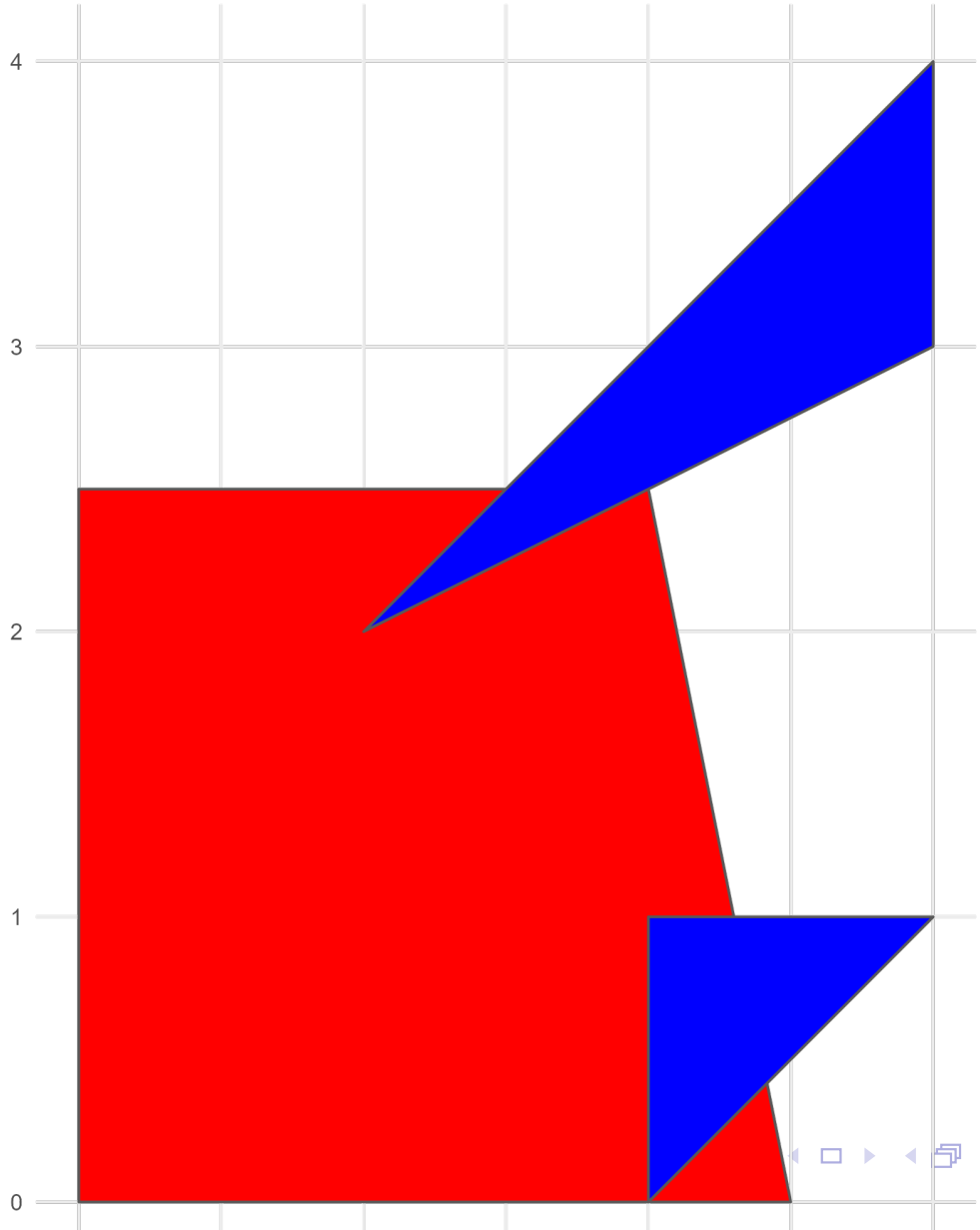
```
m1 <- rbind(c(0, 0), c(2.5, 0), c(2, 2.5), c(0, 2.5), c(0,
    0))
m2 <- rbind(c(1, 2), c(3, 3), c(3, 4), c(1, 2))
m3 <- rbind(c(2, 0), c(3, 1), c(2, 1), c(2, 0))

one_pol   <- st_polygon(list(m1))
multi_pol <- st_multipolygon(list(list(m2), list(m3)))
```

- `m1` is defined as a matrix containing five coordinate pairs that form a closed polygon: (0, 0), (2.5, 0), (2, 2.5), (0, 2.5), and (0, 0).
- **st_polygon(list(m1))** creates a single spatial polygon object using the coordinates in `m1`.
- **st_multipolygon(list(list(m2), list(m3)))** creates a multipolygon geometry by combining two polygons defined in `m2` and `m3`.

Spatial Data

Polygons



Spatial Data

Polygons

```
data_sfc <- st_sfc(one_p, multi_p,
one_l, multi_l,
one_pol, multi_pol)

class(data_sfc)
```

- `m1` is defined as a matrix containing five coordinate pairs that form a closed polygon: (0, 0), (2.5, 0), (2, 2.5), (0, 2.5), and (0, 0).
- **st_polygon(list(m1))** creates a single spatial polygon object using the coordinates in `m1`.
- **st_multipolygon(list(list(m2), list(m3)))** creates a multipolygon geometry by combining two polygons defined in `m2` and `m3`.

Spatial Features Collection

Polygons

```
data_sfc <- st_sfc(one_p, multi_p, one_l, multi_l, one_pol,
    multi_pol)
```

- **st_sfc** function combines various spatial objects, including points, linestrin and polygons.
  - This is called a spatial features collection (SFC)
- The ‘class(data_sfc)‘ command is used to determine the class or data type of the resulting spatial features collection.

Creating a Spatial Data Frame

```
data_sf <- data.frame(name = c("A", "B", "C", "D", "E", "F")
    ,
geometry = data_sfc)

data_sf <- st_sf(data_sf, geometry = data_sf$geometry)
```

- The code begins by creating a data frame called ‘data_sf‘, which has two columns: ‘name‘ and ‘geometry‘.
- The **st_sf** function is used to convert the data frame ‘data_sf‘ into a spatial data frame (sf)
  - Specifying the ‘geometry‘ column.
  - This step associates the geometrical information with the data frame, making it suitable for spatial analysis.

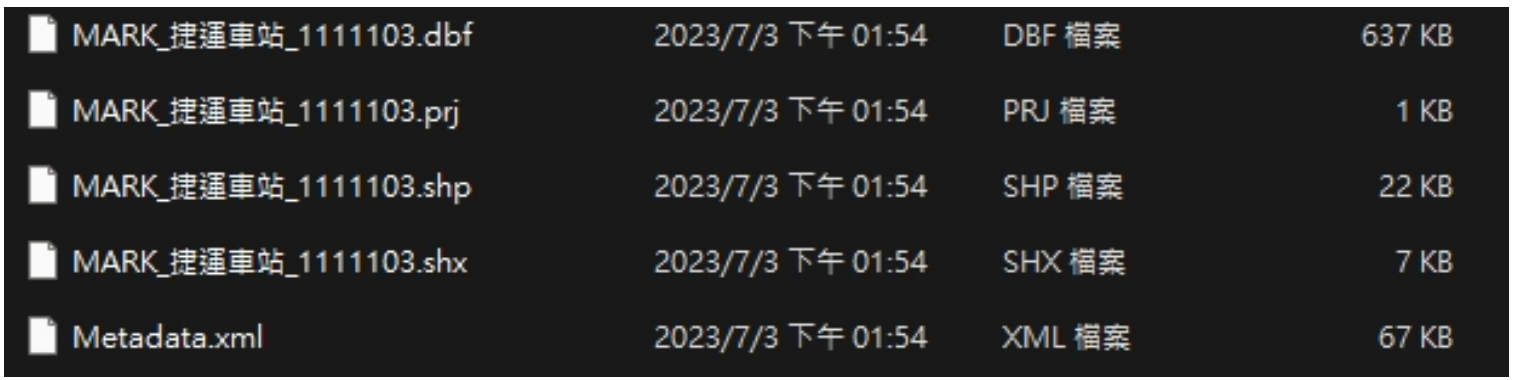
Read and Save Spatial Data

Shapefile

- A Shapefile is a common file format for storing spatial data
  - It was developed by Environmental Systems Research Institute (Esri)
  - A leading provider of GIS software and geodatabase management tools.
- Shapefiles are widely supported by GIS software, making them a versatile and interoperable choice for sharing spatial data.
- They can represent various vector-based geographic elements, such as points, lines, and polygons.
- Shapefiles consist of a set of files with the same base name but different extensions

Shapefile

- **.shp**: Record the actual coordinates of points, lines, and polygons.
- **.shx**: Constructing a spatial index for geometric elements to improve search efficiency.
- **.dbf**: Record the attribute data of geometric elements.
- **.prj**: Store the projection information of geographic coordinate systems.



Read a Shapefile

Polygon

```
town <- st_read(dsn = str_c(path, "G97_A_CALIN_P.shp"),
    options = "ENCODING=UTF-8")

town <- read_sf(str_c(path, "G97_A_CALIN_P.shp"))
```

- Both functions read spatial data
- **st_read** provides more in-depth control for advanced operations
- **read_sf** is designed for ease of use and simplicity

Read a Shapefile

Polygon

```
town_select <- read_sf(str_c(path, "G97_A_CALIN_P.shp")) %>%
select(c("SECT_NAME", "LIE_CODE", "SDFNAME"))

head(town_select)
```

- **select()**: Only select specific columns (variables) in the spatial dataset
- **head()**: show the variables in object **town_select**

Read a Shapefile

Output

```
Simple feature collection with 6 features and 3 fields
Geometry type: MULTIPOLYGON
Dimension:     XY
Bounding box:  xmin: 302583.4 ymin: 2784374 xmax: 308861.6
    ymax: 2789176
CRS:           NA
# A tibble: 6 $\times$ 4
SECT_NAME LIE_CODE   SDFNAME
    geometry
<chr>     <chr>      <chr>                          <
    MULTIPOLYGON>
1 北投區    6301200042 湖田里10鄰 (((304697.5 2786869,
    304692.4 278\ldots{}
2 士林區    6301100047 菁山里10鄰 (((307802.2 2787373,
    307831.9 278\ldots{}
3 北投區    6301200042 湖田里9鄰  (((303166.5 2786151,
    303171.5 278\ldots{}
```

Read a Shapefile

Polygon

```
1 st_geometry(town[1, ])[[1]][[1]]
```

- **st_geometry()**: This function extracts the geometry component of a simple features object
  - From the spatial object **town**, take the geometry of the first feature. Then, from that geometry, access its first primary component

Read a Shape File

Output

```
[[1]]
[,1]     [,2]
[1,] 304697.5 2786869
[2,] 304692.4 2786880
[3,] 304701.6 2786928
[4,] 304697.1 2786956
[5,] 304696.7 2786978
[6,] 304686.3 2787008
[7,] 304680.4 2787034
[8,] 304681.6 2787058
[9,] 304687.7 2787084
[10,] 304690.9 2787093
```

Project Spatial Data

Code

```
st_crs(town)

st_crs(town) <- 3826 #TWD97 / TM2 zone 121
```

- **st_crs()**: this function is used to get or set the Coordinate Reference System (CRS) of a spatial object in R
  - **3826** refers to the TWD97 / TM2 zone 121 coordinate system, which is often used for spatial data in Taiwan

Read a Shapefile

Point

```
1 mrt_station <- read_sf(str_c(path, "MARK_捷運車站_1111103.
      shp"))
2 mrt_station <- mrt_station[grep("臺北",
      mrt_station$MARKNAME1), ]
3 mrt_station <- mrt_station[-grep("出入口",
      mrt_station$MARKNAME1), ] %>% select(c("MARKNAME1"))
```

- Load a shapefile (spatial data) named "MARK_捷運車站_1111103.shp" into an object named mrt_station
  - **grep()**: filters this data to only keep stations that are located in "臺北" (Taipei)
  - **-grep()**: removes any stations that have "出入口" (entrance/exit) in their names.
  - It selects only the "MARKNAME1" column for the filtered data.

WKT Format

- Well-Known Text (WKT) is a text-based format used to represent and exchange spatial data
  - It provides a standardized way to describe geometric shapes, such as points, lines, and polygons, using a human-readable syntax.
  - It consists of text strings that define the structure and coordinates of geometries, making it platform-independent and easy to understand.

WKT Format

wkt_geom	MRTID	MRTSYS	MRTCODE	MRTTYPE	MDATE	SOURCE	DEFINITI(
MultiLineString ((303780.94179999909829348 27716	A0000000(	臺北捷運	中和新蘆線	3	201311	0	1
MultiLineString ((303769.10049999866168946 27705	A0000000(	臺北捷運	中和新蘆線	3	201311	0	1
MultiLineString ((302727.31470000091940165 27694	A0000000(	臺北捷運	中和新蘆線	3	201311	0	1
MultiLineString ((303003.29049999988637865 27684	A0000000(	臺北捷運	中和新蘆線	3	201311	0	1
MultiLineString ((304892.61319999949773774 27716	A0000000(	臺北捷運	文湖線	1	201311	0	0
MultiLineString ((304871.16460000089136884 27705(	A0000000(	臺北捷運	文湖線	1	201311	0	0
MultiLineString ((301446.31459999969229102 27710	A0000000(	臺北捷運	板南線	3	201311	0	1
MultiLineString ((302218.14130000089062378 27709	A0000000(	臺北捷運	板南線	3	201311	0	1
MultiLineString ((303769.10049999866168946 27705	A0000000(	臺北捷運	板南線	3	201311	0	1
MultiLineString ((302503.96549999935086817 27717(	A0000000(	臺北捷運	淡水信義線	3	201311	0	1
MultiLineString ((302218.14130000089062378 27709	A0000000(	臺北捷運	淡水信義線	3	201311	0	1
MultiLineString ((302727.31470000091940165 27694	A0000000(	臺北捷運	淡水信義線	3	201311	0	1
MultiLineString ((304857.12310000037541613 27695	A0000000(	臺北捷運	淡水信義線	3	201311	0	1
MultiLineString ((301979.68649999902117997 27712	A0000000(	臺灣桃園國	機場捷運	3	201606	0	1
MultiLineString ((302503.96549999935086817 27717(	A0000000(	臺北捷運	松山新店線	3	201411	0	1
MultiLineString ((303780.94179999909829348 27716	A0000000(	臺北捷運	松山新店線	3	201411	0	1

Read CSV with WKT format

Line

```
mrt_line <- read.csv(path_mrt_line) %>%
filter(MRTSYS == "Taipei Metro", MRTCODE != "")

mrt_line <- st_as_sf(mrt_line, wkt = c("wkt_geom"), crs =
    3826)
```

- This code reads a CSV file into a data frame.
  - It then filters the data to select only rows where the "MRTSYS" column equals "Taipei Metro" and the "MRTCODE" column is not empty.
- **st_as_sf()** function converts the data frame into a spatial data frame (sf).
  - The ‘wkt‘ argument specifies the name of the column containing Well-Known Text (WKT) format geometries
  - The ‘crs‘ argument sets the coordinate reference system (CRS) to 3826.

Read CSV with Coordinates

Point

```
house <- read.csv(path_house) %>%
select(c("h_price_m2", "address", "latitude", "longitude"))

house <- st_as_sf(house, coords = c("longitude", "latitude")
    ,
crs = 3826)
```

- '**st_as_sf()**' function convert the data frame into a spatial data frame (sf).
  - The 'coords' argument specifies the columns that contain the latitude and longitude coordinates, which are "longitude" and "latitude" in this case.
  - The 'crs' argument sets the coordinate reference system (CRS) to 3826.

Combine All Spatial Objects

Graph

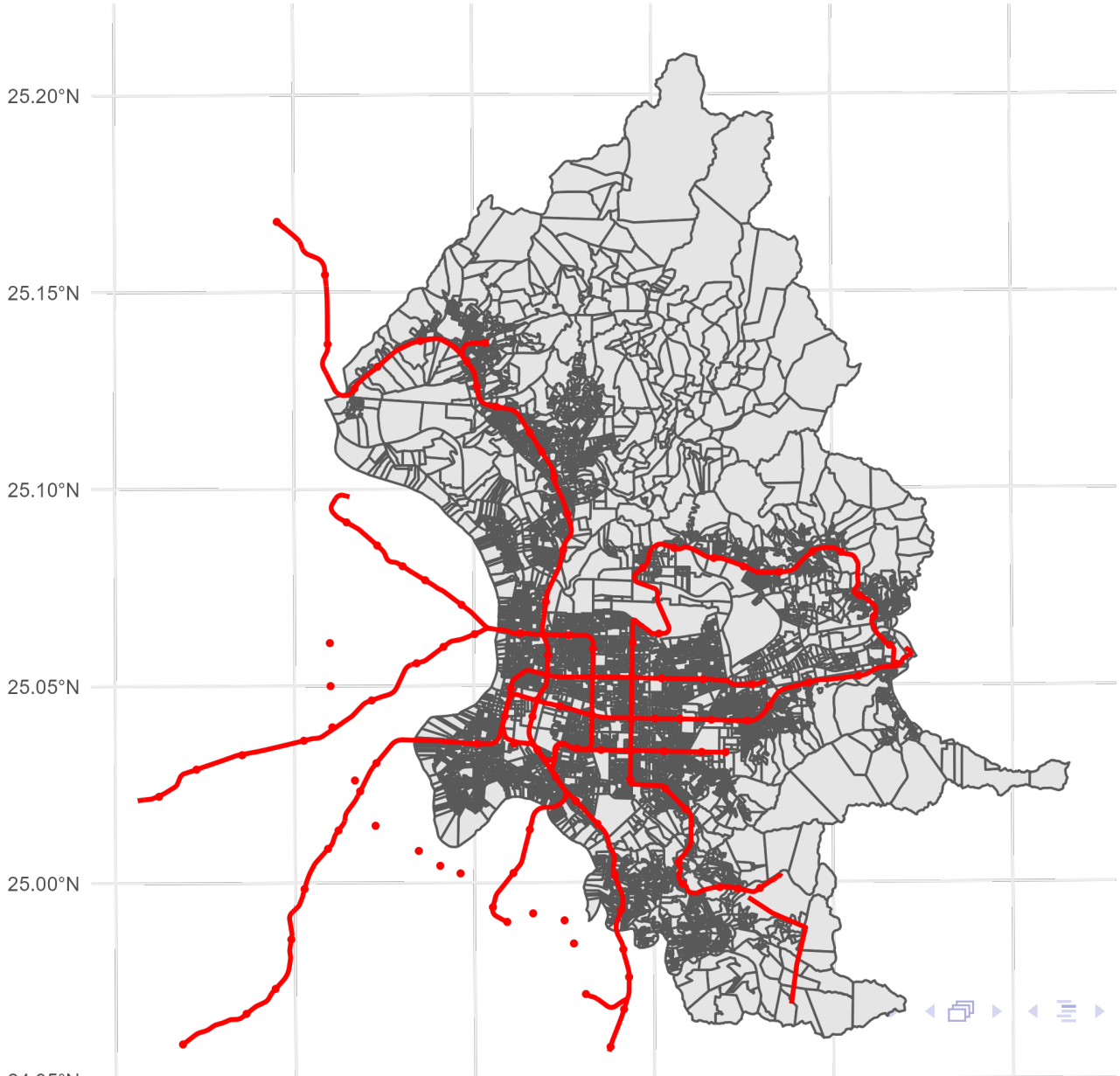
```
p<-ggplot() +
geom_sf(data = town) +
geom_sf(data = mrt_line, color = "red", size = 1) +
geom_sf(data = mrt_station, color = "red", size = 1) +
theme_minimal()

ggsave(filename = str_c(path,"tpe_mrt.png"), plot = p, width
     = 10, height = 7)
```

- This code combines multiple spatial objects in a single graph using the ‘ggplot2‘ package
  - ‘**geom_sf**‘ layers are added to the plot to display different spatial objects, including "town," "mrt_line," and "mrt_station."

Combine All Spatial Objects

MRT Line in Taipei



Re-project Spatial Data

- Sometimes we might have spatial data in different coordinate systems
  - **WGS84:** latitude-longitude coordinate system
  - **TWD97:** Taiwan Transverse Mercator coordinate system
- Therefore, it is necessary to convert them to the same coordinate system in order to perform data processing.

Re-project Spatial Data

```
st_crs(house)$epsg #TWD97 / TM2 zone 121

house_4326 <- st_transform(house, crs = 4326) #WGS84
```

- This code demonstrates the process of re-projecting spatial data
- The '**st_transform()**' function is used to re-project the 'house' spatial data to a new CRS, which is specified as 4326 (WGS84), a commonly used geographic coordinate system.
  - The CRS of re-projected data is specified as 4326 (WGS84)

Spatial Measurement – Distance

R Output

```
Simple feature collection with 6 features and 2 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 121.5111 ymin: 24.98659 xmax: 121.6171
    ymax: 25.14144
Geodetic CRS:  WGS 84
h_price_m2                                    address
                      geometry
1      161451                  台北市南港區興南街150號10樓 POINT
    (121.5987 25.05488)
2      261347 臺北市信義區安康里29鄰松勇路69巷10號17樓 POINT
    (121.5702 25.03316)
3      213985            臺北市內湖區康寧路三段187號七樓 POINT
    (121.6121 25.07005)
```

Save to a New Shapefile

```
1 write_sf(house, "house.shp")
```

- **write_sf()** function save spatial data to a new Shapefile format.
  - The second argument ‘”house.shp”‘ specifies the file name for the new Shapefile.

Drawbacks of Shapefiles

- Limited data storage:
  - Shapefiles have a maximum file size limit of 2GB and can store only up to 255 fields (attribute data)
  - Additionally, there is a restriction on attribute field names, which cannot exceed 10 characters
  - These limitations can pose constraints when working with large or complex datasets.

Drawbacks of Shapefiles

- Inefficient data storage:
  - Shapefiles store data in a binary format, which can result in larger file sizes compared to other file formats
  - Increased storage requirements and slower data access

Alternatives – Save as gpkg

- For ArcGIS user, one of the alternative data formats is GeoPackage.
- Unlike the shapefile system, it produces only a single file with .gpkg extension.
- Note that GeoPackage files can also be easily read into ArcGIS.

Alternatives – Save as GeoPackage (gpkg)

```
write_sf(house, "house.gpkg")
```

- The **write_sf()** function is used to save the ‘house‘ spatial data to a new GeoPackage file.
  - The file name ‘”house.gpkg”‘ is specified with the ‘.gpkg‘ extension, which is commonly used for GeoPackages.

```
house_gpkg <- read_sf("house.gpkg")
```

- **read_sf()** function reads the saved GeoPackage file ‘”house.gpkg”‘ back into a new spatial data frame

Alternatives – Save as RDS

- For R user, one of the alternative data formats is rds file
- The use of rds files can be particularly attractive when the dataset is large
- Because rds files are typically more memory efficient than shapefiles, eating up less of your disk memory.

Alternatives – Save as RDS (R Data)

```
1 saveRDS(house, "house.rds")
```

- The **saveRDS()** function is used to save the ‘house‘ spatial data to a new RDS file.
  - The file name ‘"house.rds"‘ is specified with the ‘.rds‘ extension, which is commonly used for R Data files.

```
1 house_rds <- readRDS("house.rds")
```

- **readRDS()** reads the saved RDS file ‘"house.rds"‘ back into a new object called ‘house_rds‘

Spatial Operations

Spatial Join

Overview

- Using geographic location to combine two spatial datasets
  - Combines attributes from one layer to another based on this geographic relationship
  - E.g., determining which address points (from one dataset) are within which neighborhoods (from another dataset)

Spatial Join

Different Types Based on Relationships

- **Intersect:** Features that overlap or share space
- **Within:** One feature geographically contained within another
- **Contains:** Opposite of "within" - one feature encloses another
- **Closest:** Joining features based on proximity

Spatial Joins

Applications

- Results used for various types of spatial analysis
- Determining density, proximity, or percentage of certain characteristics
- Analyzing number of facilities within regions, prevalence of factors near schools, etc.

Spatial Joins

R code

```
1 house_join <- st_join(house, town)
```

- This code demonstrates a spatial join operation between two spatial data frames: ‘house‘ and ‘town‘.
- Both ‘house‘ and ‘town‘ contain spatial information, such as geometries (e.g., points or polygons).
- The **st_join()** function uses the ”intersects” method, which means it joins the rows from ‘house‘ to ‘town‘ where the geometries intersect
  - Where they share any portion of space.

Spatial Joins

R Output

```
Simple feature collection with 6 features and 9 fields
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 301535 ymin: 2764405 xmax: 312256 ymax:
     2781542
Projected CRS: TWD97 / TM2 zone 121
h_price_m2                                          address
1      161451                        台北市南港區興南街150號10樓
2      261347 臺北市信義區安康里29鄰松勇路69巷10號17樓
SECT_NAME SECT_CODE LIE_NAME   LIE_CODE LI_NO          SDFKEY
1    南港區   6300900   東新里 6300900005   019
    6300900005019
2    信義區   6300200   安康里 6300200009   029
    6300200009029
SDFNAME                  geometry
1 東新里19鄰 POINT (310403 2771990)
2 安康里29鄰 POINT (307543 2769572)
```

Spatial Aggregation

Overview

- Spatial aggregation is a common procedure in GIS
  - Combining spatially located data into groups that share common geographic boundaries or characteristics

Spatial Aggregation

Advantages

- **Grouping by Geographical Boundaries:** Data points aggregated based on shared geographic areas.
- **Reducing Complexity:** Simplifies spatial data, highlighting trends or patterns.
- **Statistical Summarization:** Provides average or median values within geographical units.

Spatial Aggregation

Challenges

- **Modifiable Areal Unit Problem (MAUP):** Different aggregation levels can lead to different analysis results
  - Importance of defining appropriate spatial boundaries for aggregation.

Spatial Aggregation

R Output

```
district <- group_by(town, SECT_NAME) %>% summarise(count =
    n())
```

- This code demonstrates a spatial aggregation operation on the ‘town‘ spatial data frame.
- The **group_by()** function is used to group the data by the ‘SECT_NAME‘ variable (區名) within the ‘town‘ data frame.
  - The **summarise()** function calculates the count of features in each district

Spatial Aggregation

R Output

```
Simple feature collection with 6 features and 2 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 296266.1 ymin: 2766795 xmax: 317197.3
    ymax: 2789176
Projected CRS: TWD97 / TM2 zone 121
# A tibble: 6 $\times$ 3
SECT_NAME count                                        geometry
<chr>     <int>                                  <POLYGON [m]>
1 中山區      877 ((303114 2771127, 303073.9 2771140,
    303052.\ldots{}
2 中正區      582 ((302426.9 2768031, 302382.4 2768063,
    30237\ldots{}
3 信義區      922 ((307636.1 2767150, 307636.2 2767129,
    30763\ldots{}
```

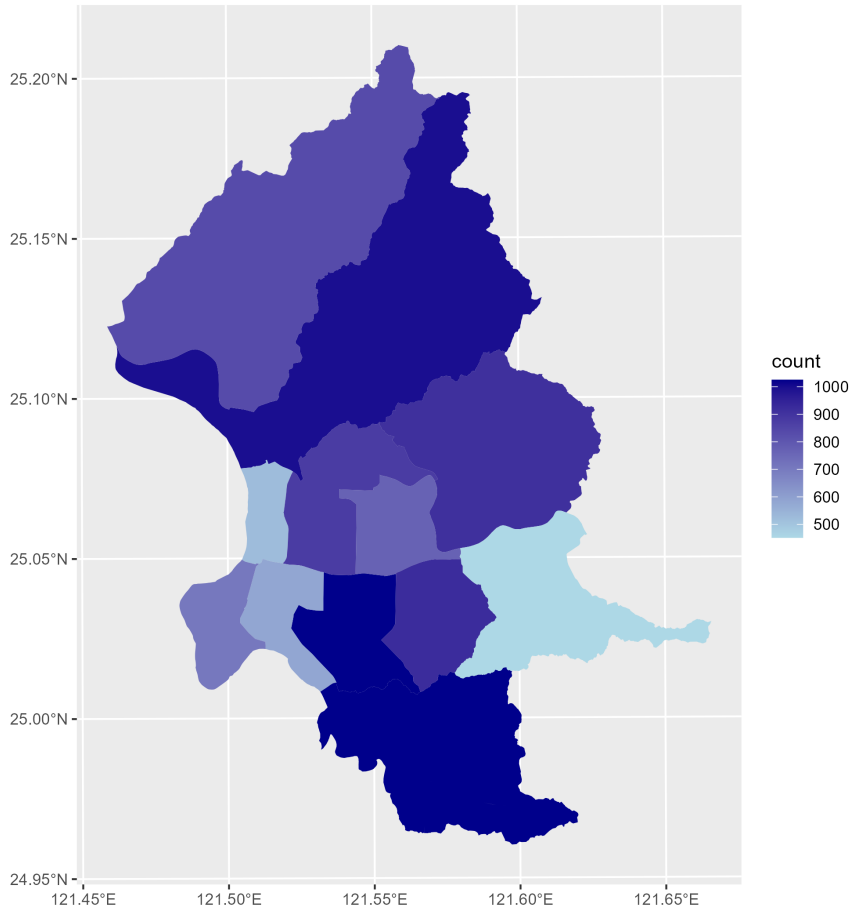
Spatial Aggregation

R Code

```
district_plot <- ggplot() +
geom_sf(data = district, aes(fill = count), linetype = "
    blank") +
scale_fill_gradient(low = "lightblue", high = "darkblue")
```

Spatial Aggregation

Housing Transactions by Districts



Spatial Aggregation

R Code

```
lie_price <- group_by(house_join, LIE_CODE) %>%
summarise(mean_price = mean(h_price_m2))
```

- This code performs spatial aggregation to calculate the mean price per square meter (‘mean_price‘) for different locations identified by the ‘LIE_CODE‘ variable (村里代碼).

Spatial Aggregation

R Output

```
Simple feature collection with 6 features and 2 fields
Geometry type: MULTIPOINT
Dimension:     XY
Bounding box:  xmin: 306087 ymin: 2771630 xmax: 307580 ymax:
    2773221
Projected CRS: TWD97 / TM2 zone 121
# A tibble: 6 $\times$ 3
LIE_CODE   mean_price                                geometry
<chr>           <dbl>                     <MULTIPOINT [m]>
1 6300100002    171924. ((306971 2773055), (306978 2773026),
    \ldots{}
2 6300100003    246851. ((306087 2772446), (306129 2772370),
    \ldots{}
3 6300100004    209986. ((306657 2772613), (306664 2772667),
    \ldots{}
```

Spatial Aggregation

R Code

```
lie_price <- st_drop_geometry(lie_price)
```

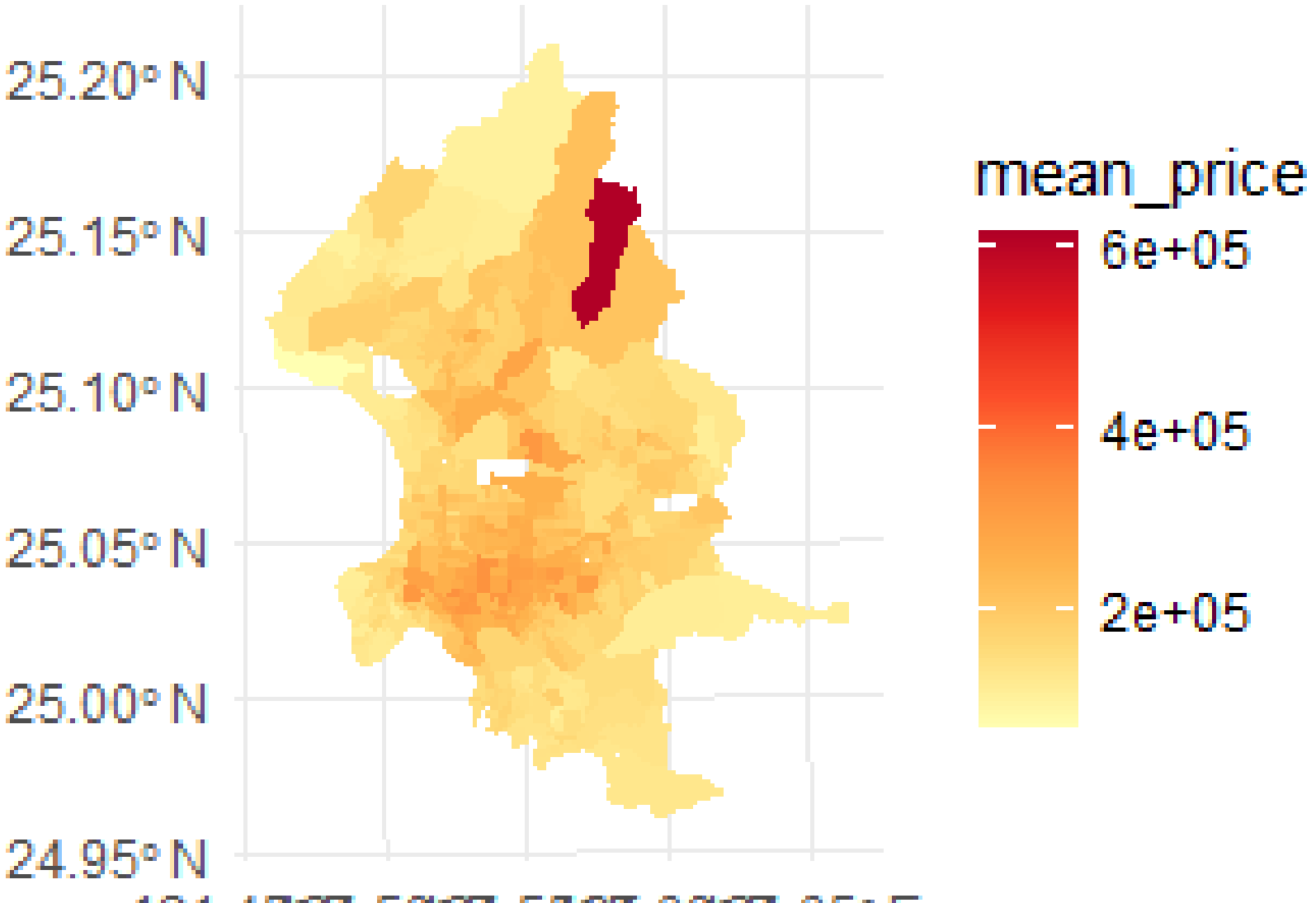
- This code removes the geometry column from the ‘lie_price‘ data frame.
  - The ‘**st_drop_geometry()**‘ function is used to drop the spatial geometry column, leaving only the non-spatial attributes in the data frame.

```
lie_price <- left_join(town, lie_price) %>% na.omit()
```

- This code performs a left join operation between the ‘town‘ spatial data frame and the modified ‘lie_price‘ data frame.
  - The ‘**left_join()**‘ function joins the two data frames based on a common key, which is an identifier related to locations.
  - ‘**na.omit()**‘ function to remove rows with missing values, if any exist in the result.

Spatial Aggregation

Housing Price by 里



Spatial Aggregation

R Output

```
# A tibble: 6 $\times$ 2
LIE_CODE   mean_price
<chr>           <dbl>
1 6300100002    171924.
2 6300100003    246851.
3 6300100004    209986.
```

Spatial Buffering

Overview

- A buffer in spatial analysis refers to a zone around a specific spatial object, measured in units of distance or time
  - **Proximity Analysis:** Identifying geographic features within a specific distance.
  - **Impact Estimation:** Assessing potential areas affected by certain events or developments.

Spatial Buffering

R Code

```
mrt_station_buffer <- st_buffer(mrt_station_taipei, 250)
```

- The '**st_buffer()**' function creates a buffer around the points in the 'mrt_station_taipei' data frame.
  - The buffer size is specified as '250', which likely represents a distance in the coordinate system's units (e.g., meters).

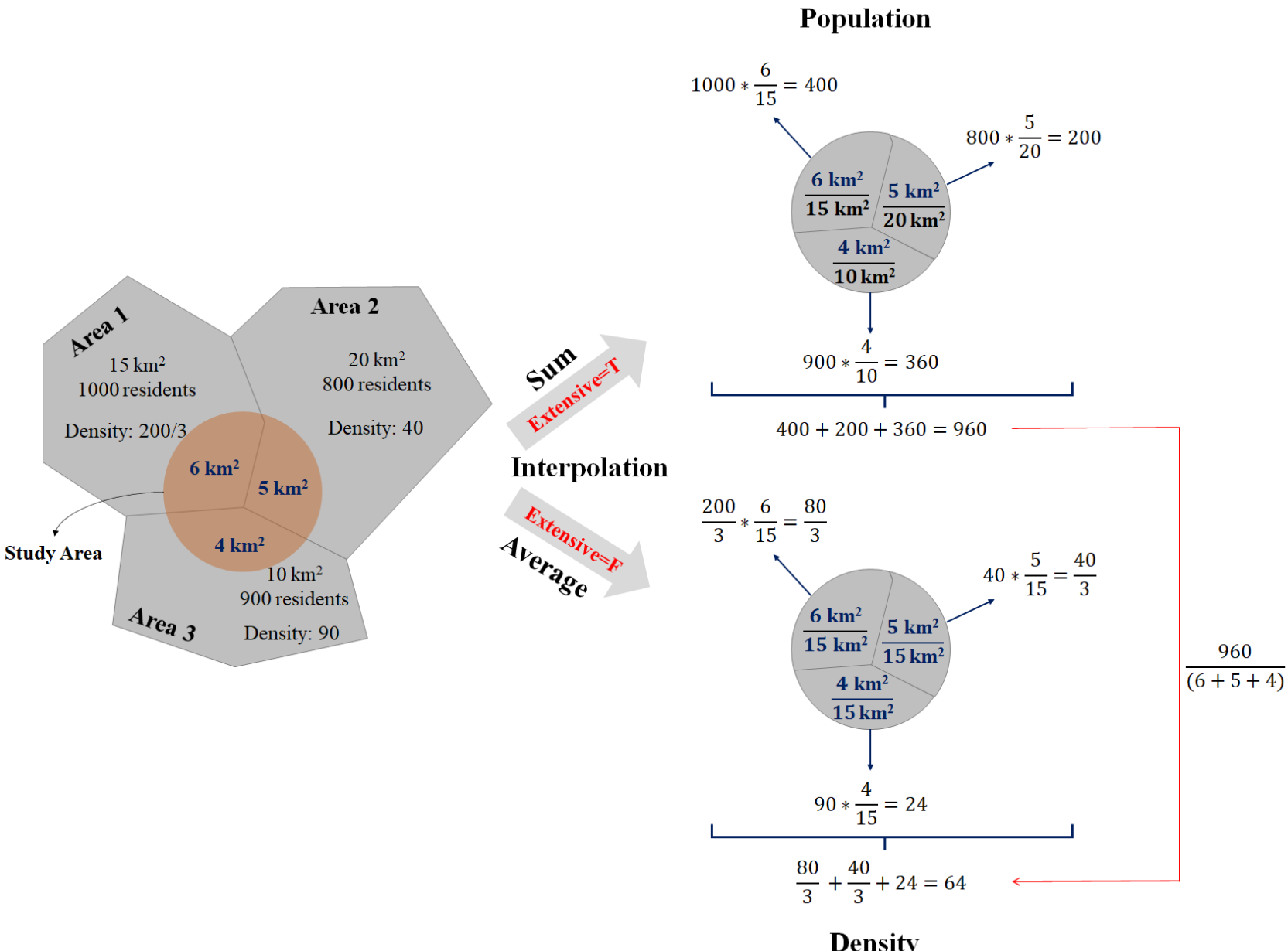
Spatial Interpolation

Overview

- Spatial interpolation is an analytical technique used in geographic studies to:
- Estimate unknown values at geographical locations where measurements are unavailable
  - Filling gaps in geographical data utilizing known data from surrounding points
  - Create continuous data representations across a landscape

Spatial Interpolation

Overview



Spatial Interpolation

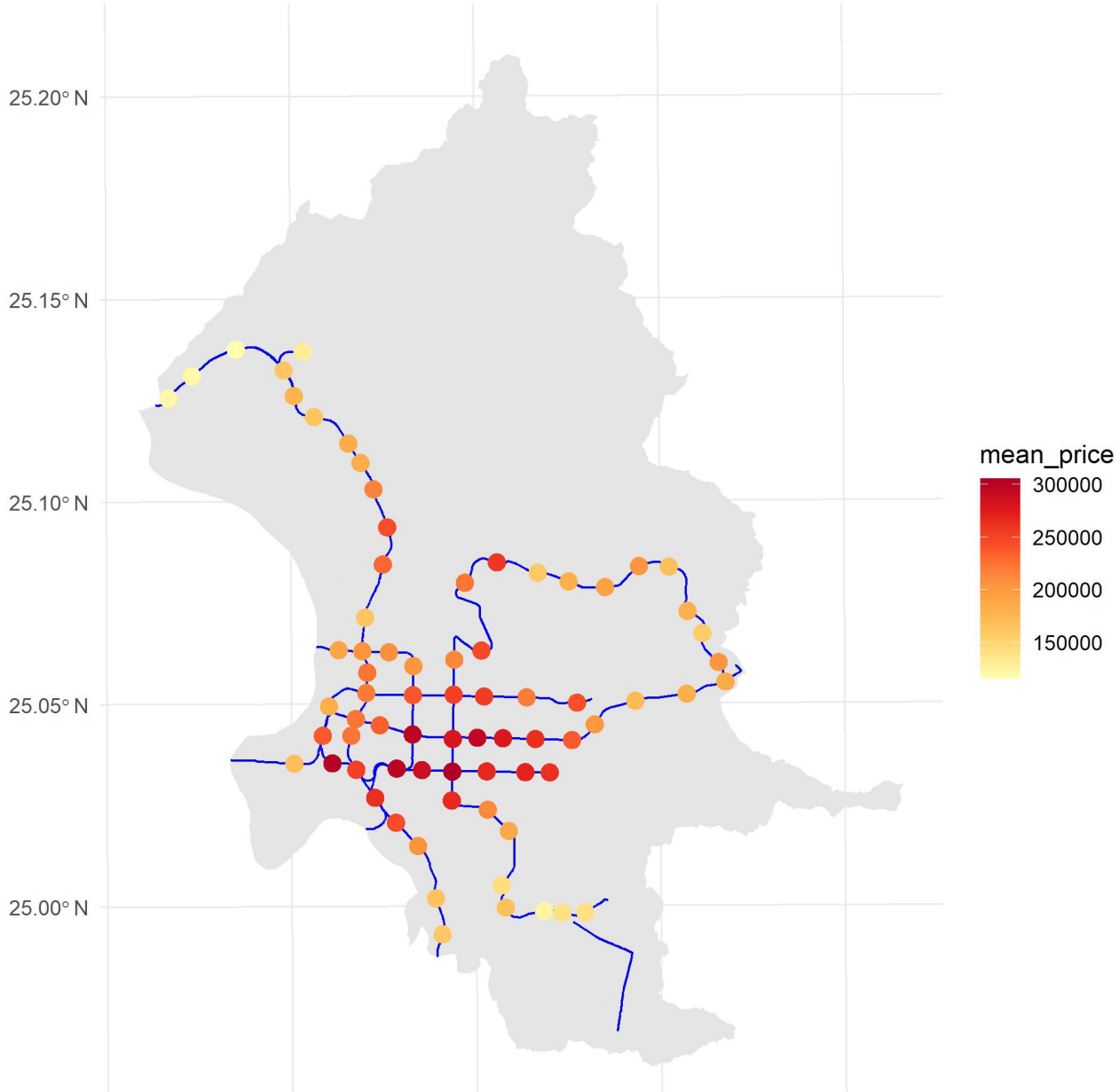
R Code

```
mrt_interpo <- st_interpolate_aw(lie_price["mean_price"],
mrt_station_buffer,
extensive = TRUE)
```

- The **st_interpolate_aw()** function performs spatial interpolation to estimate values of the `mean_price` variable from the `lie_price` data frame at the locations represented by the buffered `mrt_station_buffer`.

Spatial Interpolation

R Output



Spatial Measurement – Area

R Code

```
1 dist_area <- st_area(district)
```

- The '**st_area()**' function calculates the areas of the districts
  - It computes the area of each polygon in the 'district' data frame

Spatial Measurement – Area

R Output

```
Units: [m^2]
[1] 13918274  7540905 11242231 31943143 57381712 21953874
[7] 61078059  4768692 11341341 31249341  8681328  7450219
```

Spatial Measurement – Length

R Code

```
1 mrt_length <- st_length(mrt_line)
```

- The **st_length()** function calculates the lengths of the MRT stations.

Spatial Measurement – Length

R Output

```
Units: [m]
[1] 2473.6302 1071.9619 2219.4709 1070.2943 1133.0579
[6]  916.3198
```

Spatial Measurement – Perimeter

R Code

```
dist_border <- st_boundary(district)
```

- The '**st_boundary()**' function gets boundaries of the districts

```
dist_peri <- st_length(dist_border)
```

- The '**st_length()**' function calculates the lengths of the district boundaries.

Spatial Measurement – Perimeter

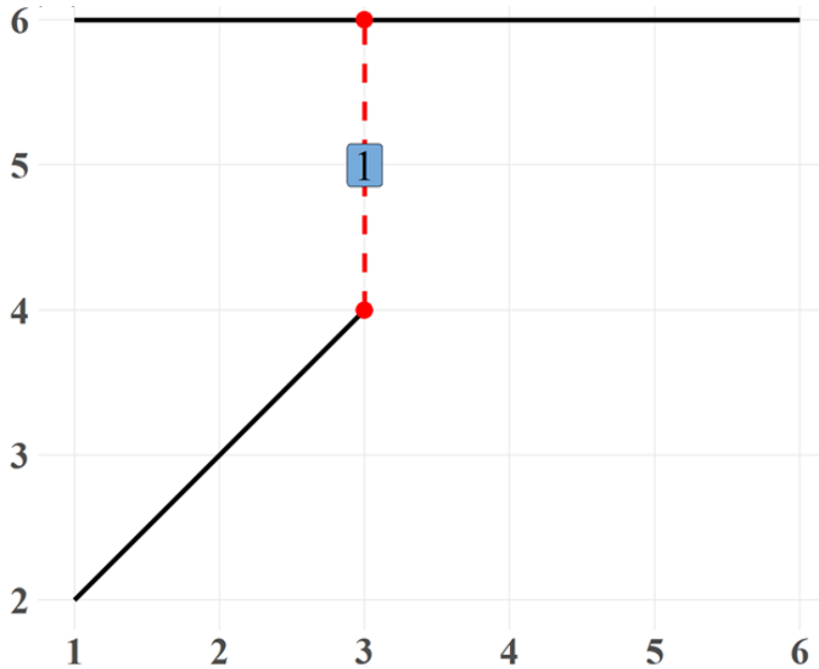
R Output

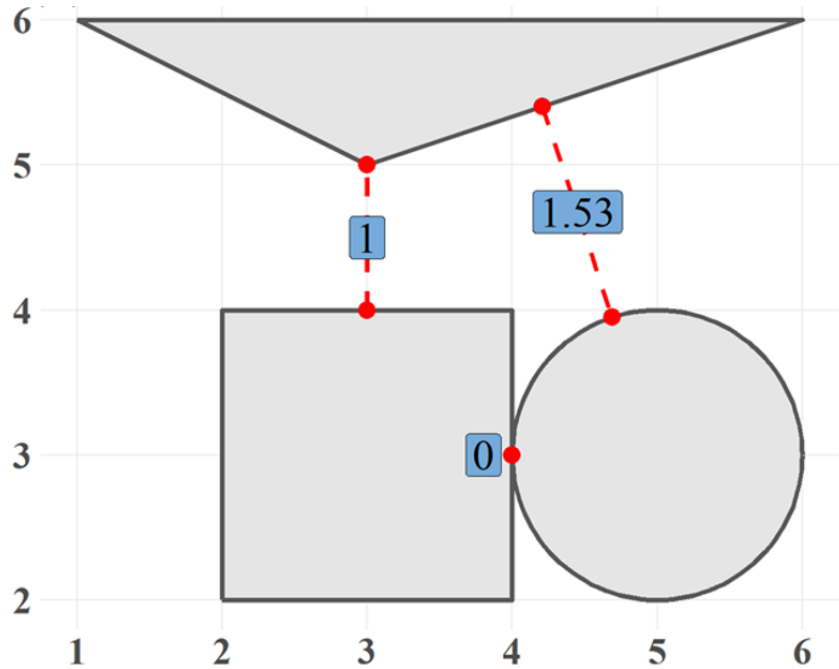
```
Units: [m]
[1] 32180.99 16395.05 30570.30 36769.93 49349.66 37274.69
[7] 85812.19 19243.09 23529.49 40296.29 20646.82 13711.05
```

Spatial Measurement – Distance

Overview

- To find the shortest distance between two geometries.





Spatial Measurement – Distance

R Code

- **st_distance(data1):** $M_{n \times n}$ - the distance between each geometry element within the geographic data
- **st_distance(data1, data2):** $M_{n \times k}$ - the distance between each paired geometry element from two geographic datasets.
- **st_distance(data1, data2, by_element=T):** $M_{1 \times n}(M_{1 \times k})$ - the distance between each pairwise matched geometry element from two geographic datasets.- $n = k$
- **st_nearest_feature(object_data, nearest_feature)**

Spatial Measurement – Distance

R Code

```
1 near_mrt <- st_nearest_feature(house, mrt_station)
```

- The '**st_nearest_feature()**' function calculates the nearest metro (MRT) station to each house location in the 'house' data frame.
  - It calculates the nearest feature (in this case, MRT station) for each house and returns an index indicating the nearest station.

Spatial Measurement – Distance

R Code

```
house$near_mrt <- mrt_station$MARKNAME1[near_mrt]

house$dis_mrt <- st_distance(house,
mrt_station[near_mrt,],
by_element = TRUE)
```

- **st_distance()** calculates the distance between each house and its nearest MRT station.
  - The '**by_element = TRUE**' argument indicates that distances should be computed for each element separately.

Spatial Measurement – Distance

R Output

```
## Simple feature collection with 6 features and 3 fields
## Geometry type: POINT
## Dimension: XY
## Bounding box: xmin: 301535 ymin: 2764405 xmax: 312256
    ymax: 2781542
## Projected CRS: TWD97 / TM2 zone 121
## address near_mrt
## 1 台北市南港區興南街150號10樓 臺北捷運昆陽站_BL21
## 2 臺北市信義區安康里29鄰松勇路69巷10號17樓 臺北捷運象山站
    _R02
## 3 臺北市內湖區康寧路三段187號七樓 臺北捷運東湖站_BR22
## dis_mrt geometry
## 1 727.47014 [m] POINT (310403 2771990)
## 2 38.06205 [m] POINT (307543 2769572)
## 3 325.02846 [m] POINT (311748 2773676)
```

Empirical Example in Economics

Empirical Example: Nathan Nunn (2008)

Nathan Nunn (2008), "**The Long-term Effects of Africa's Slave Trades**", QJE

- Does the intensity of slave trade affect economic development of African countries centuries later?

Background

- **Manning (1990, p. 124):** "Slavery was corruption: it involved theft, bribery, and exercise of brute force as well as ruses."
  - "Slavery thus may be seen as one source of precolonial origins for modern corruption."

Background

- Between 1400 and 1900, Africa experienced four simultaneous slave trades
  - Total volume of slaves traded unprecedented: $\sim$18 million
  - Slaves taken through warfare, raiding, kidnapping within Africa
- Adverse effects: social/ethnic fragmentation, political instability, weakened states

Data

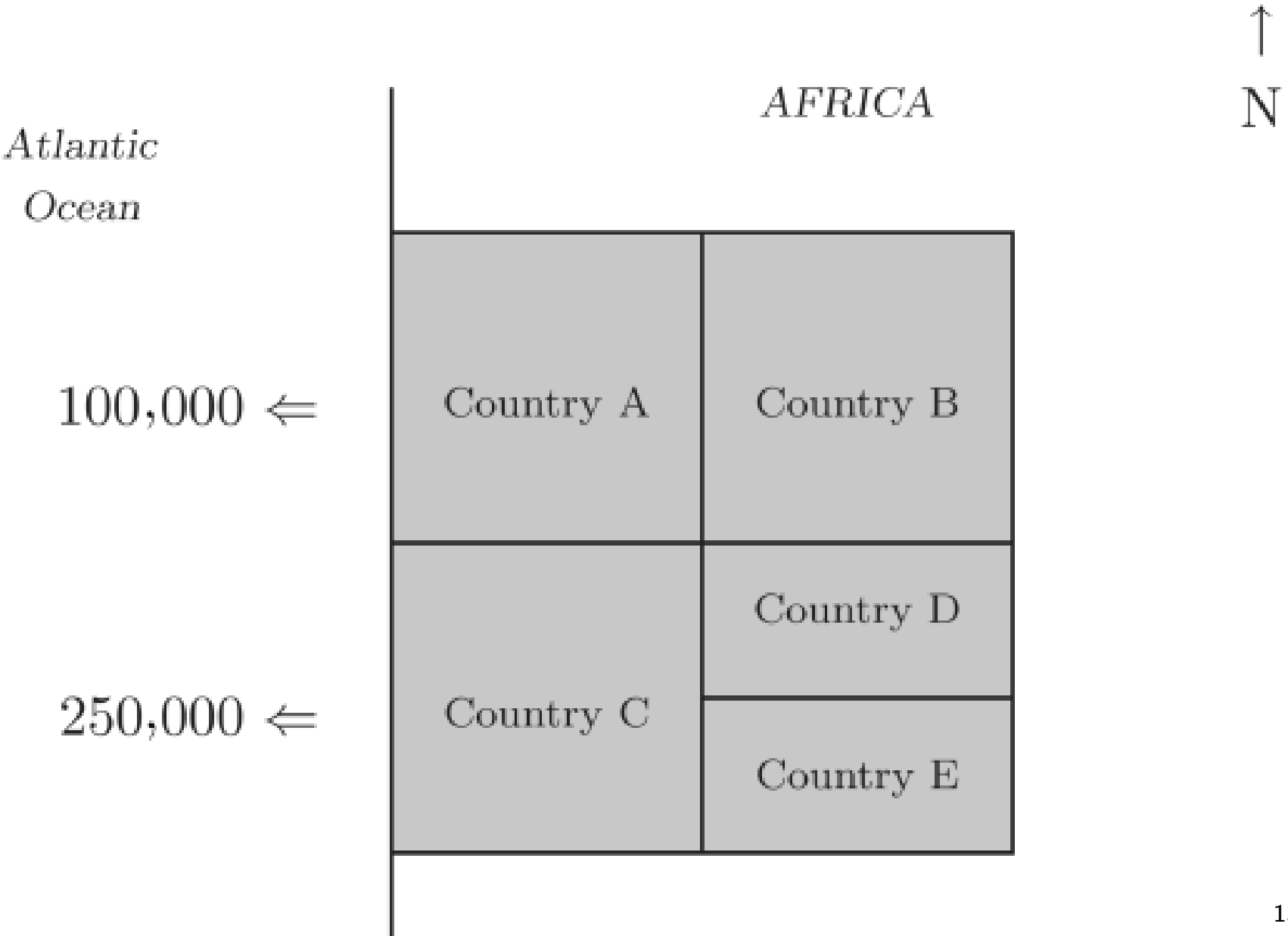
- **Raw data:**
  - Number of slaves shipped from each African country in each century between 1400 and 1900
  - Shipping records of trans-Atlantic slave trade in each African country
  - Historical documents on ethnic identities of exported slaves

- **Objective:** Accurately estimate the number of slaves shipped from each African country.
- **Challenge:** Coastal shipping data doesn't account for inland origins (e.g., slaves from Country B shipped from Country A's ports).
  - Slaves from Country B shipped from Country A's ports

- **Approach:**
  - Use shipping data to calculate initial figures from coastal countries.
  - Apply ethnicity ratios to adjust figures for landlocked countries.
- **Example:**
  - 100,000 slaves shipped from Country A
    - ★ Ratios: Country A to B is 4:1.
    - ★ Estimates: 80,000 from Country A, 20,000 each from Countries B.
  - 250,000 slaves shipped from Country C.
    - ★ Ratios: Country C to D to E is 3:1:1.
    - ★ Estimates: 150,000 from Country C, 50,000 each from Countries D and E.

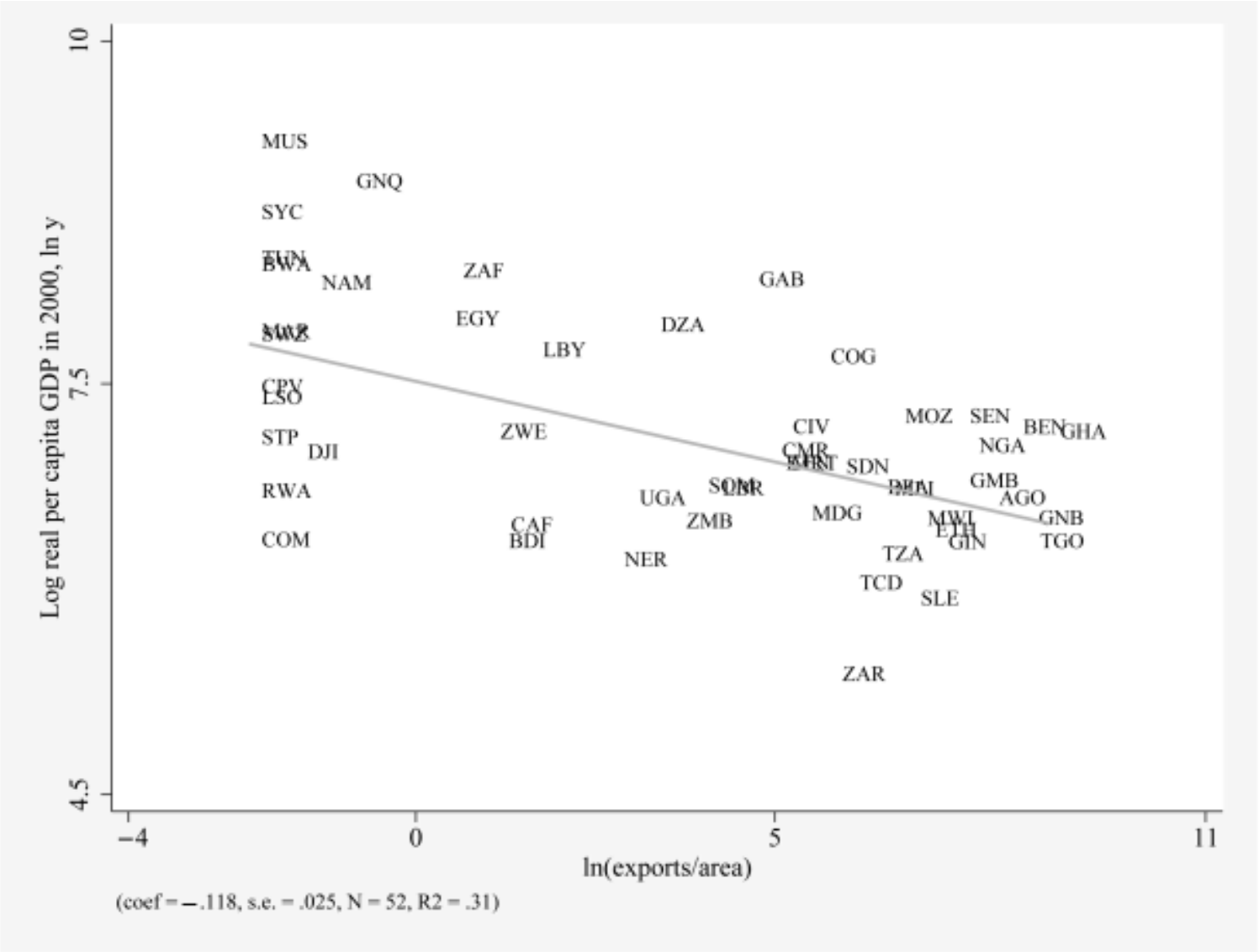
Data

Combining Shipping and Ethnicity Data



Graphical Evidence

- A robust negative relationship exists between slave exports and income.



Empirical Specification

OLS Estimation

$$\ln y_i = \beta_0 + \beta_1 \ln(\text{exports}_i/\text{area}_i) + \mathbf{C}_i\delta + \mathbf{X}_i\gamma + \epsilon_i$$

- $\ln y_i$: log of real per capita GDP in country i in 2000
- $\ln(\text{exports}_i/\text{area}_i)$: log of the total number of slaves exported between 1400 and 1900 normalized by land area
- $\mathbf{C}_i$: a vector of dummy variables that indicate the origin of the colonizer prior to independence
- $\mathbf{X}_i$: a vector of control variables that are meant to capture differences in countries' geography and climate

Results

OLS Estimates

TABLE III

RELATIONSHIP BETWEEN SLAVE EXPORTS AND INCOME

	Dependent variable is log real per capita GDP in 2000, ln y					
	(1)	(2)	(3)	(4)	(5)	(6)
ln(exports/area)	−0.112***	−0.076***	−0.108***	−0.085**	−0.103***	−0.128***
	(0.024)	(0.029)	(0.037)	(0.035)	(0.034)	(0.034)
Distance from		0.016	−0.005	0.019	0.023	0.006
equator		(0.017)	(0.020)	(0.018)	(0.017)	(0.017)
Longitude		0.001	−0.007	−0.004	−0.004	−0.009
		(0.005)	(0.006)	(0.006)	(0.005)	(0.006)
Lowest monthly		−0.001	0.008	0.0001	−0.001	−0.002
rainfall		(0.007)	(0.008)	(0.007)	(0.006)	(0.008)
Avg max humidity		0.009	0.008	0.009	0.015	0.013
		(0.012)	(0.012)	(0.012)	(0.011)	(0.010)
Avg min		−0.019	−0.039	−0.005	−0.015	−0.037
temperature		(0.028)	(0.028)	(0.027)	(0.026)	(0.025)
ln(coastline/area)		0.085**	0.092**	0.095**	0.082**	0.083**
		(0.039)	(0.042)	(0.042)	(0.040)	(0.037)
Island indicator				−0.398	−0.150	
				(0.529)	(0.516)	

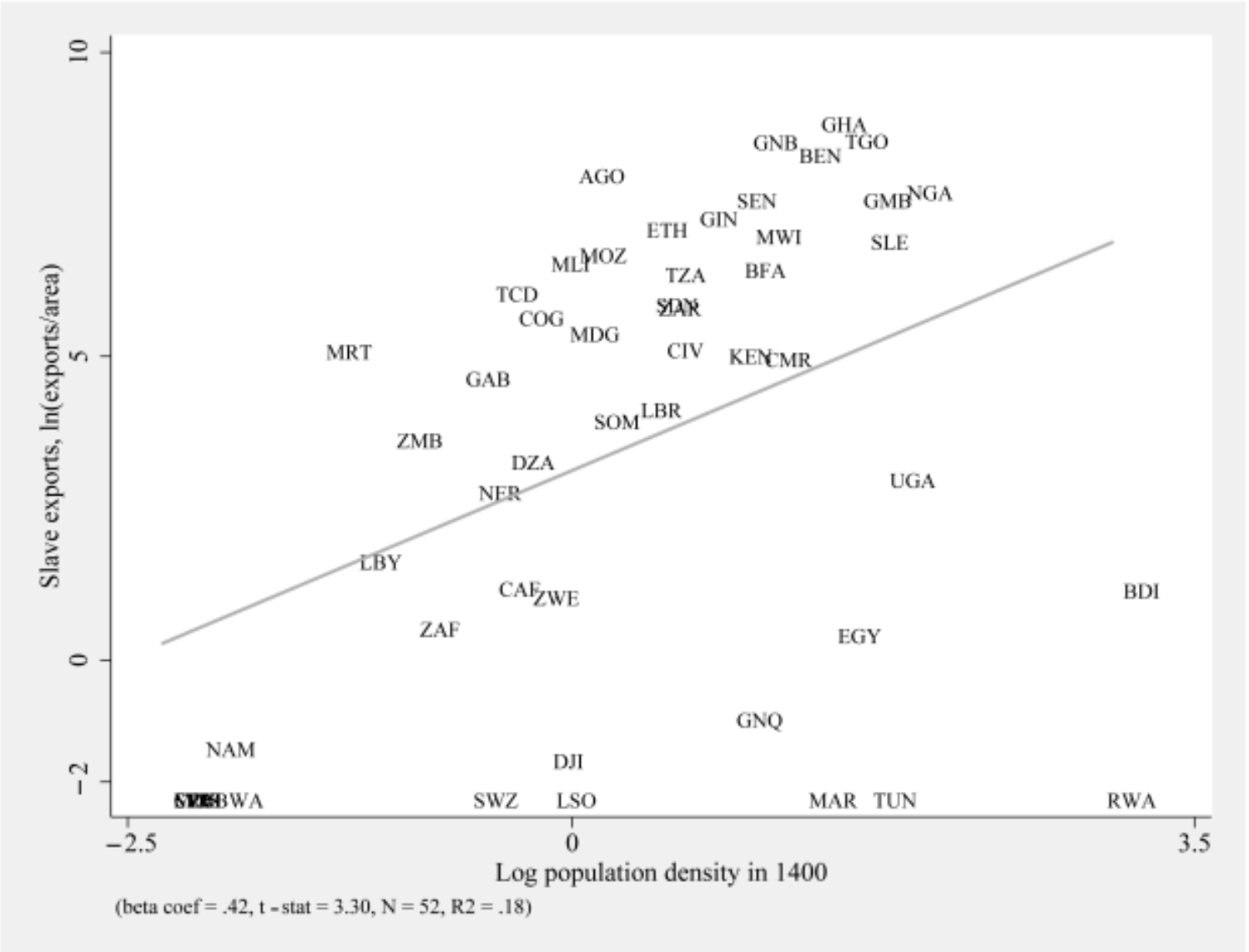
Selection versus Causation

- Two competing hypotheses:
  1. **Causation:** Slave trades had a causal negative impact on economic development
  2. **Selection:** Underdeveloped areas selected into slave trades
- Need to distinguish between the two stories.

Historic Evidence on Selection

- Initially, Europeans sought prosperous trade partners
  - Coastal raiders targeted densely populated interior states
  - Population density in 1400 positively correlated with slave exports

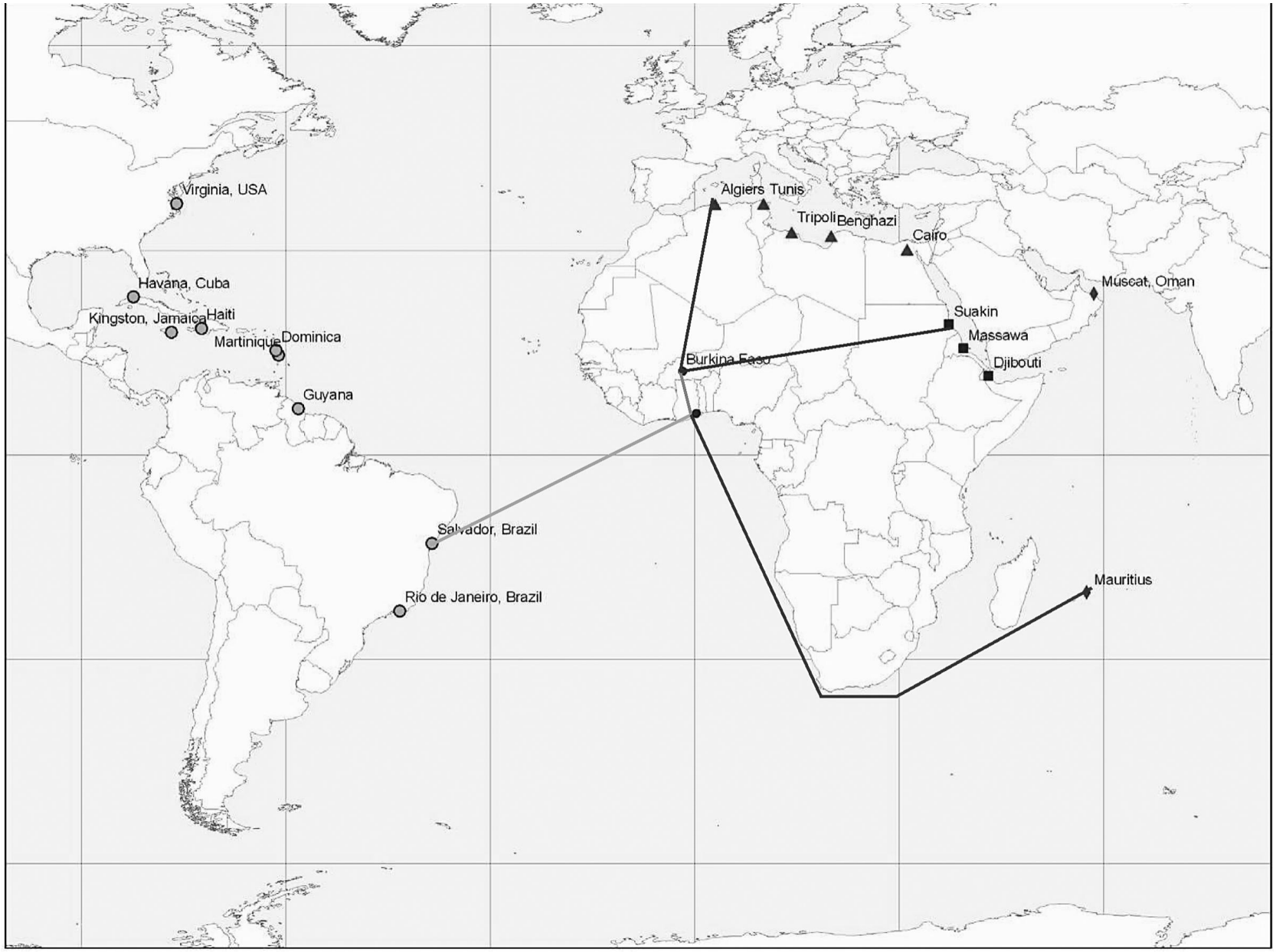
Historic Evidence on Selection



- IV for slave trade:
  - Distance to the nearest slave trade center in the Americas
    - ★ Sailing distance from the point on the coast closest to the country's centroid to the nearest trade hub

Causal Estimation

Instrumental Variable for Slave Trade



Causal Estimation

Instrumental Variable for Slave Trade

- Identifying assumptions:
  - The location of trade hubs affect the location from which the slaves are taken but not vice versa
  - Distance only affects economic development through slave trade
- Location of slave trade centers:
  - Determined by climate suitability of plantation crops/location of mines
  - Not affected by the distance to Africa
  - Distance to slave markets $\neq$ Distance to other economic opportunities

IV Estimates

First Stage

First Stage. Dependent variable is slave exports, ln(exports/area)

Atlantic distance	-1.31^{***}	-1.74^{***}	-1.32^{*}	-1.69^{**}
	(0.357)	(0.425)	(0.761)	(0.680)
Indian distance	-1.10^{***}	-1.43^{***}	-1.08	-1.57^{*}
	(0.380)	(0.531)	(0.697)	(0.801)
Saharan distance	-2.43^{***}	-3.00^{***}	-1.14	-4.08^{**}
	(0.823)	(1.05)	(1.59)	(1.55)
Red Sea distance	-0.002	-0.152	-1.22	2.13
	(0.710)	(0.813)	(1.82)	(2.40)

IV Estimates

Second Stage

Second Stage. Dependent variable is log income in 2000, $\ln y$				
ln(exports/area)	-0.208^{***}	-0.201^{***}	-0.286^{*}	-0.248^{***}
	(0.053)	(0.047)	(0.153)	(0.071)
	$[-0.51, -0.14]$	$[-0.42, -0.13]$	$[-\infty, +\infty]$	$[-0.62, -0.12]$
Colonizer fixed effects	No	Yes	Yes	Yes
Geography controls	No	No	Yes	Yes
Restricted sample	No	No	No	Yes

Main Findings

- Robust negative relationship between slave exports and income
- IV estimates confirm relationship likely causal

Mechanism

- Slave trades associated with:
  - Ethnic fractionalization and social fragmentation
  - Political instability and weakened states
- Examine if data consistent with these channels.

Conclusions

- Suggests Africa's slave trades negatively impacted long-term development
- Findings complement evidence on New World slavery and institutions
- Illuminates channel via early state underdevelopment

Implement Empirical Strategy using R

- Implementing this IV requires:
  - Identifying the country's centroid
  - Determining the closest coastal point to the centroid
  - Calculating the distance to the nearest trade hub

Implement Empirical Strategy using R

- **Datasets:**
  - Coast lines of the world
  - Country boundary in Africa
  - Boundary of ethnic regions in Africa
  - Trade hub locations

Load Spatial Data

R Code

```
#--- coast line ---#
coast <-
sf::st_read(str_c(path, "10m_coastline.shp")) %>%
st_transform(3857)

#--- African countries ---#
countries <-
sf::st_read(str_c(path, "gadm36_africa.shp")) %>%
st_transform(3857)

#--- ethnic regions ---#
ethnic_regions <-
sf::st_read(str_c(path,"borders_tribes.shp")) %>%
st_transform(3857)
```

- We first read all the GIS data we will be using in this demonstration and then reproject them to **epsg:3857**

Load Spatial Data

R Code

```
countries_simp <- rmapshaper::ms_simplify(countries)

g_countries <-
ggplot(data = countries_simp) +
geom_sf() +
theme_void()
```

- We first simplify geometries of the African countries using **rmapshaper::m**

Load Spatial Data

African Map

Find the Centroid of Each Country

R Code

```
1 countries_centroid <- st_centroid(countries)
```

- Find the centroid of each country using **st_centroid()**

Find the Centroid of Each Country

R Output

Nearest Distance between the Centroid and the Coast

R Code

```
coast_union <- st_union(coast)
minum_dist_to_coast <- st_nearest_points(countries_centroid,
    coast_union)
```

- We find its closest point on the coast line using **st_nrearest_points()**
  - **st_nearest_points(x, y)** loops through each geometry in x and returns the closest point in each feature of y

Nearest Distance between the Centroid and the Coast

R Output

Nearest Distance between the Centroid and the Coast

R Code

```
1 closest_pt_on_coast <- lwgeom::st_endpoint(
      minum_dist_to_coast)
```

- We can use **lwgeom::st_endpoint()** to extract the end point of the lines

Nearest Distance between the Centroid and the Coast

R Output

Slave Trade Center

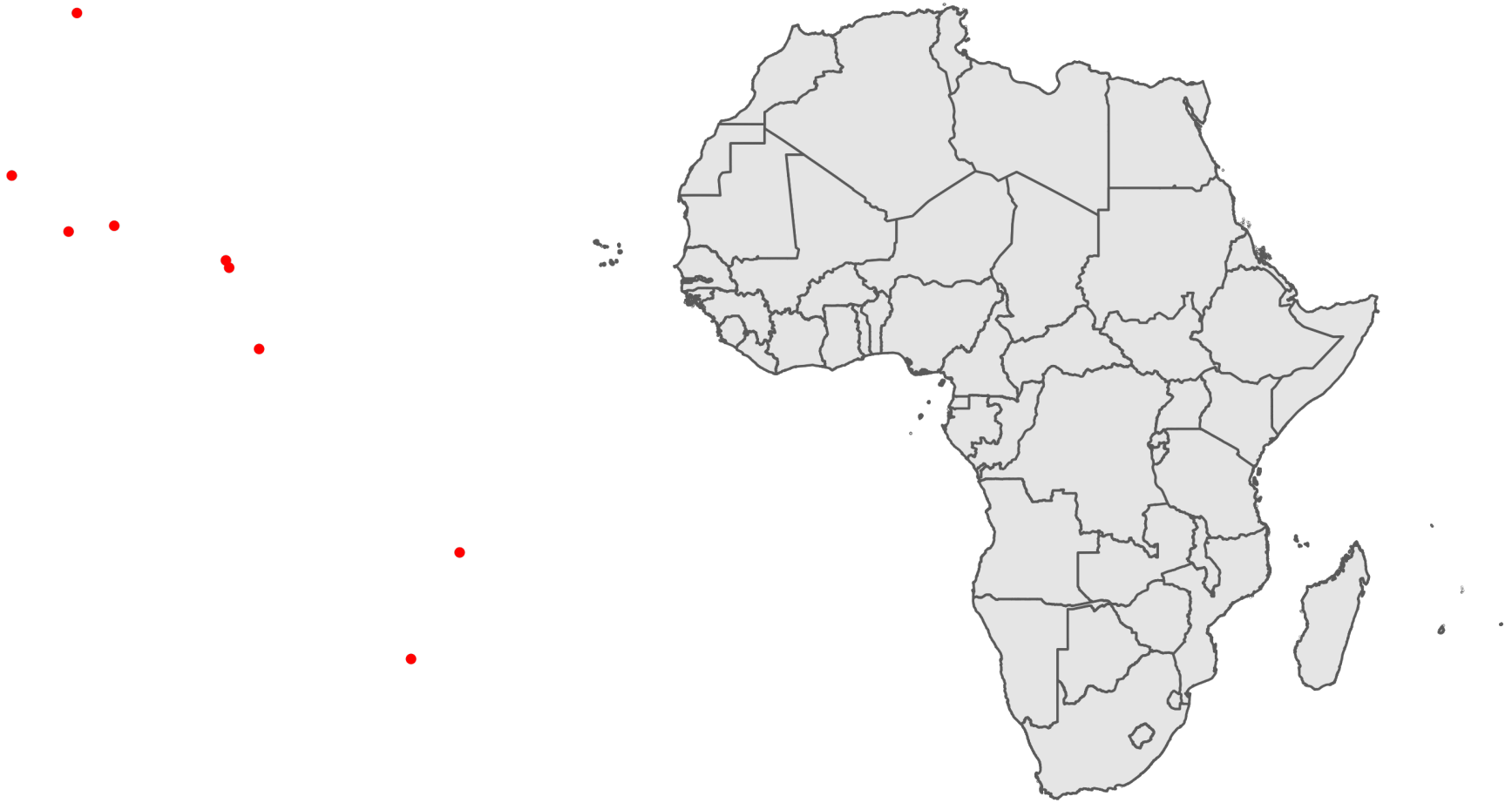
R Code

```
trade_centers_sf <-
trade_centers %>%
st_as_sf(coords = c("lon", "lat"), crs = 4326) %>%
st_transform(crs = 3857)
```

- Convert trade_centers to an sf object
- Reproejct it to epsg:3857

Slave Trade Center

R Output



Minimum Distance between Trade Center and Country

R Code

```
countries_simp$nearest_pt <- closest_pt_on_coast

trade_dist <- st_distance(countries_simp$nearest_pt,
    trade_centers_sf)

min_trade_dist <- apply(trade_dist, 1, min)
```

- Get the minimum distance between trade center and African countries

Minimum Distance between Trade Center and Country

R Output