# \*\*1 Overview\*\*

## \*\*1.1 Background\*\*

Tourism is a critical pillar of Thailand’s economy, contributing around 20% of the country’s GDP. In 2019, the sector generated approximately 90 billion US dollars from both domestic and international tourism. However, the COVID-19 pandemic severely impacted the industry, reducing revenues to 24 billion US dollars in 2020. Since September 2021, tourism revenue has shown a steady recovery, as indicated by the data from January 2019 to February 2023. Despite this rebound, the benefits of the tourism economy are concentrated in a few key provinces—Bangkok, Phuket, Chiang Mai, Sukhothai, and Phetchaburi—highlighting an uneven distribution of tourism income across the country.

## \*\*1.2 Objectives\*\*

::: panel-tabset

## Aim

Test \*\*tourism economy\*\* of Thailand are \*\*independent\*\* from space and space and time or not.

Detect where are the \*\*clusters and outliers\*\*, and the emerging \*\*hot spot/cold spot areas\*\*.

## Test

Using appropriate function of \*\*sf\*\* and \*\*tidyverse\*\*, preparing the following geospatial data layer:

- a study area layer in sf polygon features. It must be at [province level](http://localhost:3980/Take-Home\_Ex/Take-home\_Ex02/Take-home\_Ex02.html#0) (including Bangkok) of Thailand.

- a tourism economy indicators layer within the study area in sf polygon features.

- a derived tourism economy indicator layer in [spacetime s3 class of sfdep](http://localhost:3980/Take-Home\_Ex/Take-home\_Ex02/Take-home\_Ex02.html#0). Keep the time series at \*\*month and year levels\*\*.

Using the extracted data, perform

- global spatial autocorrelation analysis

- local spatial autocorrelation analysis

- emerging hotspot analysis

by using [sfdep methods](http://localhost:3980/Take-Home\_Ex/Take-home\_Ex02/Take-home\_Ex02.html#0).

Describe the spatial patterns revealed by the analysis above.

:::

# \*\*2 Data Wrangling\*\*

## \*\*2.1 Loading Packages\*\*

The following packages will be used for this exercise:

```{r}

pacman::p\_load(sf, spNetwork, tmap, tidyverse, dplyr,

spatstat, knitr, sfdep, spdep, ClustGeo,

ggpubr, cluster, factoextra, NbClust,

heatmaply, corrplot, psych, GGally, spacetime)

package\_info <- data.frame(

Package = c(

"sf", "spNetwork", "tmap", "tidyverse", "dplyr",

"spatstat", "knitr", "sfdep", "spdep", "ClustGeo",

"ggpubr", "cluster", "factoextra", "NbClust",

"heatmaply", "corrplot", "psych", "GGally", "Spacetime"

),

Description = c(

"For importing, managing, and handling geospatial data",

"For network-based spatial analysis, including kernel density estimation on networks",

"For thematic mapping and interactive visualizations",

"For non-spatial data wrangling and general data analysis with a collection of packages",

"For data manipulation and pipeline tools, part of the tidyverse",

"For spatial statistics and point pattern analysis",

"For dynamic report generation using LaTeX, HTML, and Markdown",

"For spatial statistics with a focus on spatial dependence and spatial autoregressive models",

"For spatial dependence analysis in spatial econometrics, including spatial lag and error models",

"For clustering spatial data using hierarchical clustering and geographical constraints",

"For creating publication-ready plots",

"For performing various clustering algorithms and analysis",

"For visualizing and interpreting the results of multivariate data clustering",

"For determining the optimal number of clusters in a dataset using various indices",

"For creating interactive heatmaps and hierarchical clustering visualizations",

"For visualizing correlation matrices using various methods",

"For performing descriptive statistics, factor analysis, and psychometrics",

"For visualizing correlations, pair plots, and creating enhanced data analysis plots",

"For linking a flat data set containing spatio-temporal information with the related geometry"

)

)

knitr::kable(package\_info, col.names = c("Package", "Description"))

```

## \*\*2.2 The Data\*\*

The following data are used for this study:

- Aspatial:

- \*Thailand Domestic Tourism Statistics\* at Kaggle. Version 2 is required to do the data set.

- Geospatial:

- \*Thailand - Subnational Administrative Boundaries\* at HDX.

## 2.3 Import & Preparation

:::: panel-tabset

## Aspatial

We will be importing the \*Thailand domestic tourism from 2019 to 2023\* by using the \*read\_csv()\* or \*\*readr\*\* package, and check the repeat columns with the \*duplicated()\*

```{r}

tourism <- read\_csv("data/aspatial/thailand\_domestic\_tourism\_2019\_2023\_ver2.csv")

head(tourism, 5)

any(duplicated(tourism))

```

- There are 7 variables in the \*tourism\* data, they are:

- \*\*\*Date\*\*\*: The month and year when the statistics were recorded. The dataset spans the years 2019-2023

- \*\*\*province\_thai\*\*\*: The name of the province in Thailand, in the Thai language

- \*\*\*province\_eng\*\*\*: The name of the province in Thailand, in English

- \*\*\*region\_thai\*\*\*: The name of the region in Thailand to which the province belongs, in the Thai language

- \*\*\*region\_eng\*\*\*: The name of the region in Thailand to which the province belongs, in English

- \*\*\*variable\*\*\*: The 8 type of data being recorded, such as the number of tourists or the occupancy rate.

- \*\*\*value\*\*\*: The value of the data being recorded

- We also note that values in \*variable\*  are 8 different types of data that represent different occupancy rate.

- \*\*\*no\_tourist\_all\*\*\*: The total number of domestic tourists who visited the province

- \*\*\*no\_tourist\_foreign\*\*\*: The number of foreign tourists who visited the province

- \*\*\*no\_tourist\_occupied\*\*\*: The total number of occupied hotel rooms in the province

- \*\*\*no\_tourist\_thai\*\*\*: The number of Thai tourists who visited the province

- \*\*\*occupancy\_rate\*\*\*: The percentage of occupied travel accommodation in the province

- \*\*\*revenue\_all\*\*\*: The revenue generated by the tourism industry in the province, in Thai Baht

- \*\*\*revenue\_foreign\*\*\*: The revenue generated by foreign tourists in the province, in Thai Baht

- \*\*\*revenue\_thai\*\*\*: The revenue generated by Thai tourists in the province, in Thai Baht

```{r}

filtered\_tourism <- tourism %>%

select(date, province\_eng, variable, value) %>%

pivot\_wider(names\_from = variable, values\_from = value)

head(filtered\_tourism, 10)

```

::: callout-note

Now, we have different values based on variable! Let's go to solving these data further to make it at month and year levels!

:::

```{r}

tourism\_adm1 <- filtered\_tourism %>%

rename(adm1 = province\_eng) %>%

select(adm1,

ratio\_tourist\_stay,

no\_tourist\_stay,

no\_tourist\_all,

no\_tourist\_thai,

no\_tourist\_foreign,

revenue\_all,

revenue\_thai,

revenue\_foreign,

date)

```

```{r}

head(glimpse(tourism\_adm1),5)

```

## Geospatial

```{r}

bd <- st\_read(dsn = "data/geospatial", layer = "tha\_admbnda\_adm2\_rtsd\_20220121")

# st\_crs(bd)

bd <- st\_transform(bd, crs = 32647)

# st\_crs(bd)

any(duplicated(bd))

bd\_clean <- bd %>%

select(where(~ !all(is.na(.)))) %>%

select(Shape\_Leng, Shape\_Area, ADM1\_EN, date, geometry) %>%

group\_by(ADM1\_EN) %>%

summarise(geometry = st\_union(geometry),

Shape\_Leng = sum(Shape\_Leng),

Shape\_Area = sum(Shape\_Area)) %>%

rename(adm1 = ADM1\_EN)

```

The imported shape file is simple features object of sf. From the output, we can see that there are 928 features and 19 fields, and confirm that the datum WGS 84 is correct. Moreover, based on the province level, we \*summarise()\* the geometry.

```{r}

# Calculate centroids for each province

bd\_centroid <- st\_centroid(bd\_clean)

basemap <- tm\_shape(bd\_clean) +

tm\_polygons() +

tm\_shape(bd\_centroid) +

tm\_text("adm1", size = 0.2)

tmap\_mode("plot")

basemap

```

:::

Since we have prepared the aspatial data and geospatial data, we will merge them together to help us further visualization

```{r}

merged\_data <- left\_join(tourism\_adm1, bd\_clean, by = "adm1")

```

After merging using the \*left\_join()\*, we can see there are some NA in the shape\_lengh and Shape\_area, what's more, with a further check, we find that the NA is due to a name mismatch like \*Lop Buri\* in \*\*bd\_clean and\*\* \*Lopburi\* in \*\*tourism\_adm1\*\*

```{r}

# Get unique names from both datasets

bd\_names <- unique(bd\_clean$adm1)

tourism\_names <- unique(tourism\_adm1$adm1)

# Identify names in bd\_clean not present in tourism\_adm1

mismatched\_names <- setdiff(bd\_names, tourism\_names)

mismatched\_names

```

```{r}

name\_fixes <- c(

"Buri Ram" = "Buriram",

"Chai Nat" = "Chainat",

"Chon Buri" = "Chonburi",

"Lop Buri" = "Lopburi",

"Nong Bua Lam Phu" = "Nongbualamphu",

"Phangnga" = "Phang Nga",

"Prachin Buri" = "Prachinburi",

"Si Sa Ket" = "Sisaket"

)

# Apply the fixes to the 'adm1' column in bd\_clean

bd\_clean$adm1 <- sapply(bd\_clean$adm1, function(x) {

if (x %in% name\_fixes) {

names(name\_fixes)[which(name\_fixes == x)]

} else {

x

}

})

# Perform the join again

merged\_data <- left\_join(bd\_clean, tourism\_adm1, by = "adm1")

# Remove rows with any NA values

merged\_data <- merged\_data %>% drop\_na()

```

And then we want a derived tourism economy indicator layer in spacetime s3 class of sfdep, moreover, try to keep the time series at month and year levels.

```{r}

# Check if merged\_data is already an sf object, if not convert it

if (!inherits(merged\_data, "sf")) {

merged\_data <- st\_as\_sf(merged\_data)

}

# Extract spatial part (provinces and geometry)

spatial\_data <- st\_geometry(merged\_data) # this contains the geometry

# Make sure "adm1" column corresponds to province names

# Optional: Ensure there's no mismatch in province names

# Convert 'date' column to Date type if it’s not already in that format

merged\_data$date <- as.Date(merged\_data$date, format = "%Y-%m-%d")

```

```{r}

# Extract the relevant time and spatial features

time\_data <- unique(merged\_data$date) # temporal part

# Sort the time\_data to ensure it is in increasing order

time\_data <- sort(time\_data)

```

```{r}

# Create a full combination of all spatial units (adm1) and time points (date)

spatio\_temporal\_grid <- expand.grid(

adm1 = unique(merged\_data$adm1), # Ensure these are the correct spatial units

date = unique(merged\_data$date) # Ensure this is in correct date format

)

# Merge the full spatio-temporal grid with the actual data

full\_data <- left\_join(spatio\_temporal\_grid, merged\_data, by = c("adm1", "date"))

# Check how many rows the full\_data now has

actual\_rows <- nrow(full\_data)

print(paste("Actual rows after merging:", actual\_rows))

# Replace NA values with 0 (if that's appropriate for your analysis)

full\_data[is.na(full\_data)] <- 0

```

```{r}

# Extract unique spatial data for adm1 (provinces)

spatial\_data\_unique <- merged\_data %>%

group\_by(adm1) %>%

summarise(geometry = first(geometry)) # Use the first geometry for each province

# Convert the unique spatial data to sp format

spatial\_data\_sp <- as(st\_as\_sf(spatial\_data\_unique), "Spatial")

non\_spatial\_data <- as.data.frame(full\_data)

# Remove 'adm1' (or any other conflicting columns) from non\_spatial\_data

non\_spatial\_data <- non\_spatial\_data %>%

select(-adm1) # Remove the column to avoid conflict

```

Get the spatio-temporal object

```{r}

# Create the STFDF object with the correct non-spatial data

stfdf <- STFDF(spatial\_data\_sp, time\_data, data = non\_spatial\_data)

# Optional: Print the summary of the created spatio-temporal object

summary(stfdf)

```

```{r}

saveRDS(stfdf, "data/rds/spacetime\_data.rds")

```

```{r}

# # Convert the S4 into S3

# # Extract the slots from the STFDF object

# spatial\_data <- stfdf@sp # Spatial data (spatial features)

# temporal\_data <- stfdf@time # Temporal data (time points)

# attribute\_data <- stfdf@data # Non-spatial attributes

#

# # Combine the extracted components into a list (an S3 object)

# s3\_object <- list(

# spatial = spatial\_data,

# time = temporal\_data,

# attributes = attribute\_data

# )

#

# # Check the structure of the new S3 object

# str(s3\_object)

# Extract spatial data (provinces and their geometries)

spatial\_data <- stfdf@sp # Extract spatial data (sf object)

# Extract temporal data (dates or time points)

temporal\_data <- stfdf@time

# Extract the tourism economy indicators (such as tourism revenue)

tourism\_data <- stfdf@data$revenue\_all # Replace with the relevant column name

```

```{r}

# x <- purrr::map\_dfr(1:10, ~merged\_data\_year) |>

# select(revenue\_all) |>

# mutate(

# date = sort(rep(1:10, 3458)),

# # Add noise to selected columns using random numbers

# no\_tourist\_all = no\_tourist\_all \* runif(3458, min = 0.8, max = 1.2),

# revenue\_all = revenue\_all \* runif(3458, min = 0.8, max = 1.2)

# )

# x

# Combine spatial and temporal data into an S3-compatible structure (data frame)

```

```{r}

# spt <- as\_spacetime(x, "no\_tourist\_all", "revenue\_all" , "time\_period")

```

# \*\*3 Geospatial Analysis\*\*

Since the memory limit, we will choose the 2019 Thailand data which can also ignoring the COVID-19 affection

```{r}

necessary\_columns <- c("adm1", "geometry", "Shape\_Leng", "Shape\_Area", "revenue\_all", "no\_tourist\_all" , "date")

filtered\_data <- merged\_data[, necessary\_columns]

filtered\_data\_2019 <- merged\_data[merged\_data$date == as.Date("2019-01-01"), ]

# Create a new column 'quarter' based on the date

filtered\_data\_quarters <- filtered\_data %>%

filter(format(date, "%m-%d") %in% c("03-01", "06-01", "09-01", "12-01")) %>%

mutate(quarter = case\_when(

format(date, "%m-%d") == "03-01" ~ "Q1",

format(date, "%m-%d") == "06-01" ~ "Q2",

format(date, "%m-%d") == "09-01" ~ "Q3",

format(date, "%m-%d") == "12-01" ~ "Q4"

))

filtered\_data\_Q1 <- filtered\_data\_quarters[filtered\_data\_quarters$quarter == "Q1", ]

```

## \*\*3.1 Cluster Visualization\*\*

```{r}

wm\_q <- poly2nb(filtered\_data\_2019, queen=TRUE, snap = 1)

summary(wm\_q)

```

The summary report above shows that there are 69 area units in Thailand The most connected area unit has 9 neighbours. There are two area units with only one heighbours.

```{r}

longitude <- map\_dbl(filtered\_data\_2019$geometry, ~st\_centroid(.x)[[1]])

```

```{r}

latitude <- map\_dbl(filtered\_data\_2019$geometry, ~st\_centroid(.x)[[2]])

```

```{r}

coords <- cbind(longitude, latitude)

```

```{r}

head(coords)

```

```{r}

plot(filtered\_data\_2019$geometry, border="lightgrey", main = "Neighborhood Structure", xlab = "Longitude", ylab = "Latitude")

plot(wm\_q, coords, pch = 19, cex = 0.2, add = TRUE, col= "red")

```

## \*\*3.2 Spatial Autocorrelation\*\*

We will first use Moran’s I to check whether similar values cluster geographically

```{r}

# Create a neighbors list based on contiguity (sharing borders)

nb <- poly2nb(filtered\_data\_2019)

```

```{r}

# Convert the neighbors list to a weights list

lw <- nb2listw(nb, style = "W", zero.policy = TRUE)

```

```{r}

# Perform Moran's I test for revenue\_all

moran\_test <- moran.test(filtered\_data\_2019$revenue\_all, lw)

print(moran\_test)

```

3.2 If the tourism economy is indeed spatial and spatio-temporal dependent, then, you would like to detect where are the clusters and outliers, and the emerging hot spot/cold spot areas.

# \*\*4 Geovisualisation & Geocommunication\*\*

## Using the extracted data, perform global spatial autocorrelation analysis by using sfdep methods.

## Using the extracted data, perform local spatial autocorrelation analysis by using sfdep methods.

## Using the extracted data, perform emerging hotspot analysis by using sfdep methods.