**R codes used in the analysis**

Junru Chen

10 May 2025

This file contains the main functions used for the description and the modelling of the seasonality of human parainfluenza viruses. Each funtion below is annotated where necessary. Readers who are interested in the details of each function are welcome to contact Junru Chen ([chenjr@stu.njmu.edu.cn](mailto:chenjr@stu.njmu.edu.cn)).

# 0. Prerequisites

The following R packages are needed.

library(dplyr)

library(tidyr)

library(ggplot2)

library(writexl)

library(readxl)

library(gridExtra)

library(tidyverse)

library(cowplot)

library(lemon)

library(sp)

library(gplots)

library(pROC)

library(ROCR)

library(cluster)

library(factoextra)

library(maps)

# 1. Description

## 1.1 Draw the global map of sites available

Sitemap <- ggplot() +

geom\_polygon(data = map\_data, aes(x = long, y = lat, group = group, fill = WHORegion),

color = "grey40") +

scale\_fill\_manual(values = who\_region\_colors) +

ylim(-60, 90) +

geom\_point(data = unique\_Site\_Data, aes(x = Longitude, y = Latitude, size = cut(YearTotal,

breaks = c(0, 2, 5, 8, Inf), labels = c("1-2", "3-5", "6-8", "8+"))),

shape = 21, color = "white", stroke = 0.8, alpha = 1, fill = "#cc0244") +

scale\_size\_manual(values = c("1-2" = 2.5, "3-5" = 3.5, "6-8" = 4.5, "8+" = 5.5)) +

labs(size = "Number of study-years") +

guides(size = guide\_legend(title = "Study Duration\n(Year)"),

fill = guide\_legend(title = "WHO Region")) +

geom\_hline(yintercept = c(-23.5, 23.5, -35, 35, 0), color = "black", linetype = c("dashed", "dashed", "dashed", "dashed", "solid")) +

scale\_y\_continuous(limits = c(-60, 90), breaks = c(-35, -23.5, 0, 23.5, 35, 65)) +

theme(panel.background = element\_rect(fill = "white"),

legend.title = element\_text(size=18),

legend.text = element\_text(size=18),

legend.key.size = unit(1, 'cm'),

panel.border = element\_rect(color = NA, fill = NA),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank(),

axis.title = element\_blank(),

axis.text = element\_blank()) +

geom\_text(aes(y = -23.5 + 3, x = -175, label = "-23.5°"), color = "black", size = 6) +

geom\_text(aes(y = 23.5 + 3, x = -175, label = "23.5°"), color = "black", size = 6) +

geom\_text(aes(y = -35 + 3, x = -175, label = "-35°"), color = "black", size = 6) +

geom\_text(aes(y = 35 + 3, x = -175, label = "35°"), color = "black", size = 6) +

geom\_text(aes(y = 0 + 3, x = -175, label = "0°"), color = "black", size = 6)

## 1.2 Draw the site heatmap showing monthly activity

# Function to process data for heatmap

process\_heatmap\_data <- function(data) {

# Transpose data and calculate cumulative rates

result <- lapply(1:nrow(data), function(k) {

df <- as.data.frame(t(data[k, cols]))

df$Month <- 1:nrow(df)

df %>%

arrange(desc(V1)) %>%

mutate(cumCase\_rate = cumsum(V1),

jud = ifelse(cumCase\_rate <= sum(V1) \* 0.75, 1, 0))

})

bind\_rows(result) %>%

rename(AAP = V1, SumRate = cumCase\_rate) %>%

cbind(

data.frame(Latitude = rep(data$Latitude, each = 12)),

row.names = NULL

) %>%

arrange(desc(Latitude))) %>%

mutate(

Month1 = factor(Month, levels = 1:12, ordered = TRUE),

study1 = factor(Latitude)

)

}

# Function to calculate latitude reference lines

get\_latitude\_lines <- function(data) {

c(

s35 = sum(data$Latitude <= -35),

s235 = sum(data$Latitude <= -23.5),

ns = sum(data$Latitude <= 0),

n235 = sum(data$Latitude <= 23.5),

n35 = sum(data$Latitude <= 35)

)

}

# Function to create heatmap plot

create\_heatmap <- function(data, processed\_data, title, ratio = 0.6, point\_size = 1) {

line\_positions <- get\_latitude\_lines(data)

ggplot(processed\_data, aes(Month1, study1)) +

geom\_tile(aes(fill = AAP), colour = "black", linewidth = 0.1) +

scale\_fill\_gradient2(

name = "AAP",

low = "white", mid = "white", high = "red3",

limits = c(0, 40), midpoint = 10, na.value = "red3"

) +

geom\_hline(

yintercept = line\_positions + 0.5,

linewidth = 1,

lty = c(3, 2, 1, 2, 3)

) +

geom\_point(

data = data,

aes(x = Onset, y = order(Latitude, decreasing = FALSE)),

colour = "black", shape = 3, size = point\_size, stroke = 0.5

) +

labs(x = "Month", y = "Latitude", title = title) +

coord\_fixed(ratio = ratio) +

theme(

axis.text.x = element\_text(

vjust = 0.5, hjust = 0.5, angle = 0,

face = "bold", size = 12

),

plot.title = element\_text(size = 14, hjust = 0, face = "bold"),

legend.position = "bottom",

legend.direction = "horizontal",

legend.key.width = unit(1, "cm"),

axis.text.y = element\_text(size = 14, face = "bold"),

axis.title.x = element\_text(size = 14, face = "bold"),

axis.title.y = element\_text(size = 14, face = "bold")

) +

guides(fill = "none")

}

## 1.3 Cluster analysis and visual map of countries

# Use hPIV-1 as an example

# Function to perform clustering and visualization

perform\_clustering\_and\_visualization <- function(data, country\_column, cluster\_columns, centers, title, map\_data) {

# Set the country column as row names to avoid index reset issues

data <- data.frame(Country = data[[country\_column]], data[, -which(names(data) == country\_column)])

rownames(data) <- data$Country

data <- data[, -1]

# Select and standardize the cluster columns

cluster\_data <- data[, cluster\_columns]

cluster\_data <- scale(cluster\_data) # Standardize the data

# Determine the optimal number of clusters using the elbow method

fviz\_nbclust(cluster\_data, kmeans, method = "wss")

# Perform k-means clustering

km <- kmeans(cluster\_data, centers = centers, nstart = 25)

print(km) # Print the clustering results

# Add the cluster assignments back to the original data

data$Cluster <- km$cluster

# Prepare the map data

map\_data <- map\_data("world")

# Convert cluster to integer for visualization

map\_data$Cluster <- as.integer(map\_data$Cluster)

# Create the cluster map

cluster\_map <- ggplot(data = map\_data, aes(x = long, y = lat)) +

geom\_map(data = map\_data, map = map\_data, aes(map\_id = region, x = long, y = lat),

color = "black", size = 0.25) +

geom\_hline(yintercept = c(-23.5, 23.5, -35, 35, 0), color = "black",

linetype = c("dashed", "dashed", "dashed", "dashed", "solid")) +

scale\_y\_continuous(limits = c(-60, 90), breaks = c(-35, -23.5, 0, 23.5, 35, 65)) +

theme(panel.border = element\_rect(color = "black", fill = NA),

panel.grid.major = element\_blank(),

panel.grid.minor = element\_blank()) +

geom\_map(data = map\_data, map = map\_data, aes(map\_id = region, fill = factor(Cluster)),

color = "black", size = 0.25) +

scale\_fill\_manual(values = c("#2878b5", "#9ac9db", "#f8ac8c", "#c82423", "#ff8884"), na.value = "lightgrey") +

labs(fill = "Cluster", title = title, x = "Longitude", y = "Latitude")

# Print the map

print(cluster\_map)

}

# Perform clustering and visualization for HPIV1

perform\_clustering\_and\_visualization(

data = newdata\_HPIV1\_cou,

country\_column = "Country",

cluster\_columns = c("Peak", "Onset"),

centers = 3,

title = "hPIV-1 Cluster",

map\_data = map\_data

)

# 2. Modelling work

## 2.1 Attaching monthly average meteorological data

For this part, please refer to the R package “GSODR” in this section. First, the function nearest\_stations is needed to identify the nearest weather stations of each site. Secondly, the function get\_GSOD is needed to extract the weather data.

## 2.2 Model comparison

#Find the optimal parameter span and meteorological predictor that maximises the AUC in each model and calcuate AUC values (using example of hPIV-1)

# Define a function to perform the analysis for different climate variables

analyze\_climate\_variable <- function(data, variable\_name, span\_values, flag\_column = "flag", aap\_column = "AAP") {

auc\_values <- numeric()

# Loop through each span value

for (span\_val in span\_values) {

# Initialize columns for predictions and normalized predictions

data$Prc\_Percent <- NA

data$Prc\_Percent\_Normalized <- NA

for (i in 1:nrow(data)) {

train <- data[-i, ]

test <- data[i, ]

# Fit a loess model using the specified variable

model <- loess(reformulate(variable\_name, response = aap\_column), data = train, span = span\_val)

data$Prc\_Percent[i] <- predict(model, newdata = test)

data$Prc\_Percent[is.na(data$Prc\_Percent)] <- 0

}

# Normalize the predictions within each Study\_ID group

data <- data %>%

group\_by(Study\_ID) %>%

mutate(Prc\_Percent\_Normalized = Prc\_Percent / sum(Prc\_Percent) \* 100)

# Calculate cumulative sums for each Study\_ID

data$cum\_sum\_j <- 0

for (id in unique(data$Study\_ID)) {

subset\_data <- subset(data, Study\_ID == id)

for (j in 1:12) {

sorted\_values <- sort(subset\_data$Prc\_Percent\_Normalized, decreasing = TRUE)

result <- sum(sorted\_values[1:j])

data$cum\_sum\_j[data$Study\_ID == id & data$Prc\_Percent\_Normalized == sorted\_values[j]] <- result

}

}

roc\_res <- roc(data[[flag\_column]], data$cum\_sum\_j)

auc\_values <- c(auc\_values, roc\_res$auc)

}

# Return a data frame with span values and corresponding AUC values

return(data.frame(span = span\_values, AUC = auc\_values))

}

# Define a function to analyze multiple combinations of climate variables

analyze\_climate\_combinations <- function(data, combinations, span\_values, flag\_column = "flag", aap\_column = "AAP") {

results <- list()

for (combination in combinations) {

formula <- reformulate(combination, response = aap\_column)

auc\_values <- numeric()

for (span\_val in span\_values) {

data$Prc\_Percent <- NA

for (i in 1:nrow(data)) {

train <- data[-i, ]

test <- data[i, ]

model <- loess(formula, data = train, span = span\_val)

data$Prc\_Percent[i] <- predict(model, newdata = test)

data$Prc\_Percent[is.na(data$Prc\_Percent)] <- 0

}

data <- data %>%

group\_by(Study\_ID) %>%

mutate(Prc\_Percent\_Normalized = Prc\_Percent / sum(Prc\_Percent) \* 100)

data$cum\_sum\_j <- 0

for (id in unique(data$Study\_ID)) {

subset\_data <- subset(data, Study\_ID == id)

for (j in 1:12) {

sorted\_values <- sort(subset\_data$Prc\_Percent\_Normalized, decreasing = TRUE)

result <- sum(sorted\_values[1:j])

data$cum\_sum\_j[data$Study\_ID == id & data$Prc\_Percent\_Normalized == sorted\_values[j]] <- result

}

}

roc\_res <- roc(data[[flag\_column]], data$cum\_sum\_j)

auc\_values <- c(auc\_values, roc\_res$auc)

}

results[[paste(combination, collapse = "\_")]] <- data.frame(span = span\_values, AUC = auc\_values)

}

return(results)

}

# Define a function to find the maximum AUC values across multiple data frames

find\_max\_auc <- function(data\_frames) {

all\_max\_rows <- data.frame(matrix(ncol = 0, nrow = 0))

# Loop through each data frame

for (name in names(data\_frames)) {

df <- data\_frames[[name]]

df <- na.omit(df, select = "AUC") # Remove rows with NA in AUC column

max\_value <- max(df$AUC) # Find the maximum AUC value

max\_row <- df[df$AUC == max\_value, ] # Get the row with the maximum AUC value

max\_row$Data\_Frame\_Name <- name # Add the data frame name

all\_max\_rows <- rbind(all\_max\_rows, max\_row)

}

# Order the results by AUC in descending order

all\_max\_rows <- all\_max\_rows[order(-all\_max\_rows$AUC), ]

return(all\_max\_rows)

}

# Define span values

span\_values <- seq(0.05, 0.95, by = 0.05)

# Analyze single climate variables

single\_variables <- c("mc\_TEMP", "mi\_TEMP", "mc\_RH", "mc\_PRCP")

results\_single <- lapply(single\_variables, function(var) {

analyze\_climate\_variable(HPIV1\_Mete, var, span\_values)

})

# Analyze combinations of climate variables

combinations <- list(c("mi\_TEMP", "mc\_RH"), c("mi\_TEMP"), c("mi\_TEMP", "mc\_PRCP"))

results\_combinations <- analyze\_climate\_combinations(HPIV1\_Mete, combinations, span\_values)

# Combine all results into a single list

all\_results <- c(results\_single, results\_combinations)

# Find the maximum AUC values across all results

max\_auc\_results <- find\_max\_auc(all\_results)

## 2.3 Finalised model fitting with plots

# Use temperate hPIV-1 as an example

Te1 <- ggplot(NTe\_1, aes(x = NTe\_1$mi\_TEMP,

y = NTe\_1$mc\_RH)) +

geom\_point(aes(color = NTe\_1$Prc\_Percent\_Normalized )) +

labs(x = "cTEMP",y = "RH",color = "AAP",title = "A.hPIV-1 in temperate region") +

scale\_color\_gradientn(colors = c("#3f5efb", "#fbd786", "#ed213a"),

values = c(0, 0.33, 0.66, 1),

limits = c(0, 30),

oob = scales::squish) +

geom\_vline(xintercept = 0, linetype = "solid", size = 0.8) +

geom\_hline(yintercept = 0, linetype = "solid", size = 0.8)+

# scale\_x\_continuous(limits = c(-10, 10), breaks = c(-10,-5, 0, 5,10)) +

# scale\_y\_continuous(limits = c(-10, 15), breaks = seq(-10, 15, 5)) +

theme(panel.background = element\_rect(fill = "white", colour = "black"),

panel.grid.major = element\_line(colour = "grey95"),

panel.grid.minor = element\_line(colour = "grey95"))