```r
# --------------------------
# Sample Script: Directed DTW Analysis for Individuals and Groups
# Load required R packages
# --------------------------

library(tidyverse)
library(dtw)
library(reshape2)
library(qgraph)

# --------------------------
# Create 6 example patient datasets (3 items, EDE_A to EDE_C)
# --------------------------

Patient1 <- tibble(time = 1:10,
  EDE_A = c(3, 2, 3, 2, 4, 4, 6, 5, 4, 3),
  EDE_B = c(2, 3, 3, 2, 1, 3, 5, 6, 5, 4),
  EDE_C = c(5, 4, 5, 6, 4, 5, 3, 4, 2, 3))

Patient2 <- tibble(time = 1:10,
  EDE_A = c(1, 2, 3, 2, 3, 4, 3, 4, 5, 4),
  EDE_B = c(1, 1, 2, 3, 3, 3, 4, 4, 5, 5),
  EDE_C = c(4, 5, 5, 4, 3, 4, 3, 2, 3, 4))

Patient3 <- tibble(time = 1:10,
  EDE_A = c(4, 3, 4, 5, 6, 5, 4, 2, 4, 5),
  EDE_B = c(3, 4, 3, 4, 5, 6, 5, 4, 2, 4),
  EDE_C = c(1, 2, 2, 0, 3, 2, 3, 4, 3, 2))

Patient4 <- tibble(time = 1:10,
  EDE_A = c(2, 2, 3, 4, 4, 3, 1, 3, 4, 3),
  EDE_B = c(1, 2, 2, 3, 4, 4, 3, 0, 3, 4),
  EDE_C = c(6, 5, 4, 5, 4, 2, 0, 2, 0, 1))

Patient5 <- tibble(time = 1:10,
  EDE_A = c(5, 1, 5, 4, 5, 4, 3, 1, 3, 4),
  EDE_B = c(4, 5, 1, 1, 3, 5, 4, 3, 2, 3),
  EDE_C = c(2, 3, 2, 1, 5, 4, 5, 4, 3, 2))

Patient6 <- tibble(time = 1:10,
  EDE_A = c(3, 4, 3, 2, 3, 4, 5, 4, 3, 2),
  EDE_B = c(2, 3, 4, 3, 2, 3, 4, 5, 4, 3),
  EDE_C = c(5, 6, 5, 3, 5, 6, 5, 4, 5, 6))

# --------------------------
# Functions to calculate directed DTW distances
# --------------------------

calc_pair_directed <- function(var1, var2, db, timewindow = 1) {
  # Calculate the directed dynamic time warping (DTW) distance
  # between two time series: var2 (follower) and var1 (leader)
  distance <- dtw(
    db[[var2]],      # Time series that follows (var2)
    db[[var1]],      # Time series that leads (var1)
    window.size = timewindow,  # Allow a maximum lag/lead within this time window
    step.pattern = symmetric2, # Allow symmetric steps (flexible matching both forward and
backward slightly)
    # Define a custom window restriction:
    window.type = function(iw, jw, window.size, ...) {
      # Only allow alignment if var2 lags behind or is at most 'window.size' ahead of var1
      return((iw - jw) <= window.size & iw - jw >= 0) } ) %>%
    .$normalizedDistance  # Extract the normalized distance from the DTW output
  # That is the total accumulated DTW distance divided by the length of the warping path
  return(distance) # Return the calculated directed distance
}

calc_directed_matrix <- function(db, timewindow = 1, individual = TRUE) {
  # Define the number of interpolation points between original timepoints
  no_interpolations <- 5
  # Interpolate each time series to reduce alignment errors
  # at the beginning and end of the trajectories
  db_interpolated <- map(db, ~ approx(.x, n = nrow(db) * no_interpolations)$y) %>%
    bind_cols()
  # Create all possible (Var1, Var2) item combinations
```

```
  pair_list <- crossing(Var1 = names(db), Var2 = names(db)) %>%
    # For each pair, calculate the directed dynamic time warping distance
    mutate(value = map2_dbl(Var2, Var1, calc_pair_directed, db_interpolated,
      timewindow * no_interpolations))
  # Turn the pairwise distance values into a square matrix (items × items)
  distance_matrix1 <- acast(pair_list, Var1 ~ Var2, value.var = "value")
  # Calculate directedness: (distance A→B - distance B→A) normalized by their sum
  distance_matrix2 <- (distance_matrix1 - t(distance_matrix1)) /
    (distance_matrix1 + t(distance_matrix1) + 1e-9)
  # Set all negative directed values to 0 (keep only positive directed relations)
  if (individual) { directed_distance <- ifelse(distance_matrix2 < 0, 0, distance_matrix2)}
  else {directed_distance <- distance_matrix2}
  # Return the directed distance matrix
  return(directed_distance)
}

# --------------------------
# Calculating the directed distance for individual patient
# --------------------------

# Patient 1: calculate directed matrix and plot, and remove time column
patient1_directed <- calc_directed_matrix(Patient1[-1], timewindow = 1, individual = TRUE)
qgraph(patient1_directed, threshold = .2) # Arrow only if directed distance is greater than 0.2

# Patient 2: calculate directed matrix and plot
patient2_directed <- calc_directed_matrix(Patient2[-1])
qgraph(patient2_directed, threshold = .2) # Arrow only if directed distance is greater than 0.2
# --------------------------
# Visualize the directed DTW network for the group of 6 patients
# --------------------------

# The base R function scale(x) returns a matrix, not a simple numeric vector
scale2 <- function(x, na.rm = TRUE) (x - mean(x, na.rm = na.rm)) / sd(x, na.rm)

# Combine individual patient datasets and transform into long format
tidy_patients <- bind_rows(
  Patient1 %>% mutate(id = 1),
  Patient2 %>% mutate(id = 2),
  Patient3 %>% mutate(id = 3),
  Patient4 %>% mutate(id = 4),
  Patient5 %>% mutate(id = 5),
  Patient6 %>% mutate(id = 6)) %>%
  # Reshape from wide (EDE_A, EDE_B, EDE_C) to long format (item, score)
  pivot_longer(cols = starts_with("EDE_"), names_to = "item", values_to = "score") %>%
  # Standardize scores within each item
  group_by(item) %>%
  mutate(score = scale2(score)) %>%
  ungroup()

directed_group <- tidy_patients %>%
  # Spread back to wide format (columns = items)
  pivot_wider(names_from = item, values_from = score) %>%
  # Remove the time variable (we only keep item scores)
  select(-time) %>%
  # Group data by patient ID
  group_by(id) %>%
  # Nest the data per patient (one mini-dataset per patient)
  nest() %>%
  # For each patient, calculate the directed distance matrix
  mutate(distance = map(data, ~ calc_directed_matrix(.x, timewindow = 1,
    individual = FALSE))) %>%
  # Keep only the patient ID and the calculated distance
  select(id, distance)

# Create directed distance matrices for each patient
directed_group <- tidy_patients %>%
  pivot_wider(names_from = item, values_from = score) %>%  # Spread back to wide format
  select(-time) %>%  # Remove the time point column
  group_by(id) %>%
  nest() %>%  # Nest the data per patient
  mutate(distance = map(data, ~calc_directed_matrix(.x, timewindow = 1,
    individual = FALSE))) %>%  # Calculate directed matrices
  select(-data)

# Aggregate directed matrices across patients
```

```r
directed_network <- directed_group %>%
  # Extract only the list-column 'distance' (each patient's distance matrix)
  pull(distance) %>%
  # Melt each distance matrix into long format (Var1, Var2, value)
  map(melt) %>%
  # Combine all melted matrices into one big long-format table
  bind_rows() %>%
  # Group the data by each unique edge (Var1 -> Var2)
  group_by(Var1, Var2) %>%
  # Summarise: run a t-test for each edge across patients
  summarise(ttest = list(t.test(value)),      # Store full t-test output
    .groups = "drop") %>%                     # Ungroup automatically after summarise
  # Extract p-values and mean differences from the t-test results
  mutate(p = map_dbl(ttest, "p.value"),       # Extract p-value
    mean_diff = map_dbl(ttest, "estimate"),   # Extract mean directed distance
    # Set mean_diff to 0 if the edge is non-significant or negative
    mean_diff = ifelse(p > 0.05 | mean_diff < 0, 0, mean_diff)) %>%
  # Reshape the long table back into a directed matrix
  acast(Var1 ~ Var2, value.var = "mean_diff")

# Visualize the group-level directed DTW network
qgraph(directed_network)
```