Take-Home Assignment 2

David Chiam

2025-05-22

# **Take Home Assignment 2 on Mini-Challenge 3**

## **1 - Getting Started**

### 1.1 - Load the R packages for this assignment

For the purpose of this assignment, five R packages will be used. They are tidyverse, jsonlite, tidygraph, ggraph and SmartEDA.

In the code chunk below, p\_load() of pacman package is used to load the R packages into R environment.

pacman::p\_load(tidyverse, jsonlite,   
 tidygraph, ggraph, SmartEDA,   
 ggrepel, scales, lubridate, dplyr, viridis)

### 1.2 - Importing Knowledge Graph Data

For the purpose of this exercise, *mc3.json* file will be used. Before getting started, you should have the data set in the **data** sub-folder.

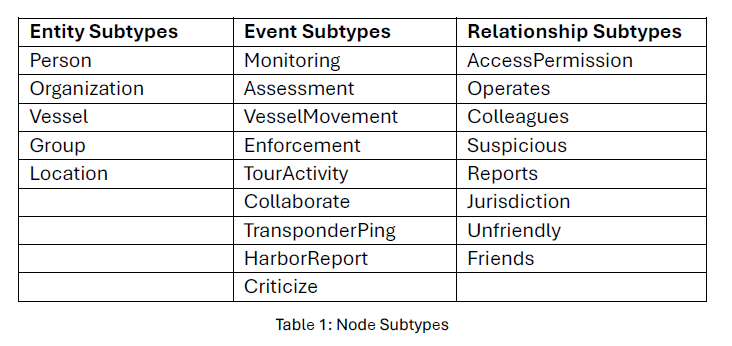
In the code chunk below, fromJSON() of **jsonlite** package is used to import *mc3.json* file into R and save the output object

MC3 <- fromJSON("data/MC3\_graph.json")  
MC3\_schema <- fromJSON("data/MC3\_schema.json")

### 1.3 - **Data Overview**

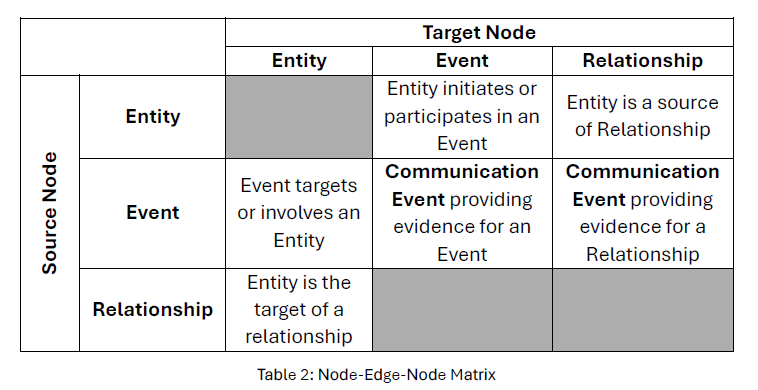
The dataset was provided by VAST Challenge for MC3. This report utilizes two core datasets: MC3\_graph.json, which encodes the knowledge graph of communications, events, and relationships; and MC3\_schema.json, which defines the structure, subtypes, and attributes of each node and edge type within the graph. There ngraph contains a total of 1159 nodes and 3226 edges. Full description of node attributes and edge attributes is shown below.

**Nodes Attributes** are as such:



Node Subtypes

**Edge Attributes** are as such:



Node-Edge-Node Matrix

### 1.4 - **Inspecting knowledge graph structure**

Before preparing the data, it is always a good practice to examine the structure of *mc3* knowledge graph.

In the code chunk below glimpse() is used to reveal the structure of *mc3* knowledge graph.

glimpse(MC3)

|  |
| --- |
| Warning |
| Notice that *Industry* field is in list data type. In general, this data type is not acceptable by tbl\_graph() of **tidygraph**. In order to avoid error arise when building tidygraph object, it is wiser to exclude this field from the edges data table. However, it might be still useful in subsequent analysis. |

### 1.5 - Extract the edges and nodes tables

Next, as\_tibble() of **tibble** package package is used to extract the nodes and links tibble data frames from *mc3* tibble dataframe into two separate tibble dataframes called *mc3\_nodes* and *mc3\_edges* respectively.

mc3\_nodes <- as\_tibble(MC3$nodes)  
mc3\_edges <- as\_tibble(MC3$edges)

### 1.6 - Brief Analyis of the extracted edges and nodes data (visuals)

It is time for us to apply appropriate EDA methods to examine the data.

**Nodes:**

### The Code Chunk

ExpCatViz(data=mc3\_nodes,  
 col="lightblue")

### The Plots

ExpCatViz(data=mc3\_nodes,  
 col="lightblue")

|  |
| --- |
| Note |
| What useful discovery can you obtained from the visualisation above? |

**Edges:**

### The Code Chunk

ExpCatViz(data=mc3\_edges,  
 col="lightblue")

### The Plots

|  |
| --- |
| Note |
| What useful discovery can you obtained from the visualisation above? |

## 2 - **Data Cleaning and Wrangling**

### **2.1 - Cleaning and wrangling nodes**

Code chunk below performs the following data cleaning tasks:

* convert values in id field into character data type,
* exclude records with id value are na,
* exclude records with similar id values,
* exclude thing\_collected field, and
* save the cleaned tibble dataframe into a new tibble datatable called mc3\_nodes\_cleaned.

mc3\_nodes\_cleaned <- mc3\_nodes %>%  
 mutate(id = as.character(id)) %>%  
 filter(!is.na(id)) %>%  
 distinct(id, .keep\_all = TRUE) %>%  
 select(-thing\_collected)

### **2.2 - Cleaning and wrangling edges**

Next, the code chunk below will be used to:

* rename source and target fields to from\_id and to\_id respectively,
* convert values in from\_id and to\_id fields to character data type,
* exclude values in from\_id and to\_id which not found in the id field of mc3\_nodes\_cleaned,
* exclude records whereby from\_id and/or to\_id values are missing, and
* save the cleaned tibble dataframe and called it mc3\_edges\_cleaned.

mc3\_edges\_cleaned <- mc3\_edges %>%  
 rename(from\_id = source,   
 to\_id = target) %>%  
 mutate(across(c(from\_id, to\_id),   
 as.character)) %>%  
 filter(from\_id %in% mc3\_nodes\_cleaned$id,   
 to\_id %in% mc3\_nodes\_cleaned$id) %>%  
 filter(!is.na(from\_id), !is.na(to\_id))

Next, code chunk below will be used to create mapping of character id in mc3\_nodes\_cleaned to row index.

node\_index\_lookup <- mc3\_nodes\_cleaned %>%  
 mutate(.row\_id = row\_number()) %>%  
 select(id, .row\_id)

Next, the code chunk below will be used to join and convert from\_id and to\_id to integer indices. At the same time we also drop rows with unmatched nodes.

mc3\_edges\_indexed <- mc3\_edges\_cleaned %>%  
 left\_join(node\_index\_lookup,   
 by = c("from\_id" = "id")) %>%  
 rename(from = .row\_id) %>%  
 left\_join(node\_index\_lookup,   
 by = c("to\_id" = "id")) %>%  
 rename(to = .row\_id) %>%  
 select(from, to, is\_inferred, type) %>%  
 filter(!is.na(from) & !is.na(to))

Next the code chunk below is used to subset nodes to only those referenced by edges.

used\_node\_indices <- sort(  
 unique(c(mc3\_edges\_indexed$from,   
 mc3\_edges\_indexed$to)))  
  
mc3\_nodes\_final <- mc3\_nodes\_cleaned %>%  
 slice(used\_node\_indices) %>%  
 mutate(new\_index = row\_number())

We will then use the code chunk below to rebuild lookup from old index to new index.

old\_to\_new\_index <- tibble(  
 old\_index = used\_node\_indices,  
 new\_index = seq\_along(  
 used\_node\_indices))

Lastly, the code chunk below will be used to update edge indices to match new node table.

mc3\_edges\_final <- mc3\_edges\_indexed %>%  
 left\_join(old\_to\_new\_index,   
 by = c("from" = "old\_index")) %>%  
 rename(from\_new = new\_index) %>%  
 left\_join(old\_to\_new\_index,   
 by = c("to" = "old\_index")) %>%  
 rename(to\_new = new\_index) %>%  
 select(from = from\_new, to = to\_new,   
 is\_inferred, type)

### **2.3 - Building the tidygraph object**

Now we are ready to build the tidygraph object by using the code chunk below.

mc3\_graph <- tbl\_graph(  
 nodes = mc3\_nodes\_final,  
 edges = mc3\_edges\_final,  
 directed = TRUE  
)

After the tidygraph object is created, it is always a good practice to examine the object by using str().

str(mc3\_graph)

## 3 - **Exploratory Data Analysis (after cleaning & wrangling)**

Several of the **ggraph** layouts involve randomisation. In order to ensure reproducibility, it is necessary to set the seed value before plotting by using the code chunk below.

set.seed(1818)

### **3.1 - Visualising the knowledge graph**

Shows how many nodes are of type Entity, Event, or Relationship.

mc3\_nodes\_final %>%  
 count(type, sort = TRUE) %>%  
 ggplot(aes(x = reorder(type, -n), y = n, fill = type)) +  
 geom\_col() +  
 geom\_text(aes(label = n), vjust = -0.3) +  
 labs(title = "Node Type Distribution", x = "Type", y = "Count") +  
 theme\_minimal() +  
 theme(legend.position = "none")

In the code chunk below, **ggraph** functions are used to create the whole graph.

ggraph(mc3\_graph,   
 layout = "fr") +  
 geom\_edge\_link(alpha = 0.3,   
 colour = "gray") +  
 geom\_node\_point(aes(color = `type`),   
 size = 4) +  
 geom\_node\_text(aes(label = type),   
 repel = TRUE,   
 size = 2.5) +  
 theme\_void()

### **3.2 - Entity Sub\_type Distribution**

Focuses on what kinds of actors are in the graph — Person, Vessel, Organization, etc.

# Define color mapping  
subtype\_colors <- c(  
 "Person" = "#2ca5ff",  
 "Organization" = "#f5ee15",  
 "Vessel" = "#FB7E81",  
 "Group" = "#25e158",  
 "Location" = "#ec4bff"  
)  
  
mc3\_nodes\_final %>%  
 filter(type == "Entity") %>%  
 count(sub\_type, sort = TRUE) %>%  
 ggplot(aes(x = reorder(sub\_type, n), y = n, fill = sub\_type)) +  
 geom\_col() +  
 coord\_flip() +  
 geom\_text(aes(label = n), hjust = -0.1) +  
 labs(title = "Entity Sub-type Distribution", x = "Sub-type", y = "Count") +  
 theme\_minimal() +  
 theme(legend.position = "none")

### **3.3 - Event Sub\_type Distribution**

To understand what kinds of actions dominate — Communication, Monitoring, Assessment, etc.

mc3\_nodes\_final %>%  
 filter(type == "Event") %>%  
 count(sub\_type, sort = TRUE) %>%  
 ggplot(aes(x = reorder(sub\_type, n), y = n, fill = sub\_type)) +  
 geom\_col() +  
 coord\_flip() +  
 geom\_text(aes(label = n), hjust = -0.1) +  
 labs(title = "Event Sub-type Distribution", x = "Sub-type", y = "Count") +  
 theme\_minimal() +  
 theme(legend.position = "none")

### **3.4a - List of Communication Participants**

This finds all Entities that **sent or received** communication events — i.e., actors who participated in messaging.

library(DT)  
  
# Step 1: Get all Communication Event IDs  
comm\_event\_ids <- mc3\_nodes\_cleaned %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 pull(id)  
  
# Step 2: Extract 'sent' edges for communication events  
comm\_sent\_edges <- mc3\_edges\_cleaned %>%  
 filter(type == "sent", to\_id %in% comm\_event\_ids) %>%  
 select(comm\_id = to\_id, sender\_id = from\_id)  
  
# Step 3: Extract 'received' edges for same communication events  
comm\_received\_edges <- mc3\_edges\_cleaned %>%  
 filter(type == "received", from\_id %in% comm\_event\_ids) %>%  
 select(comm\_id = from\_id, receiver\_id = to\_id)  
  
# Step 4: Join sent and received edges by communication ID  
comm\_pairs <- comm\_sent\_edges %>%  
 inner\_join(comm\_received\_edges, by = "comm\_id")  
  
# Step 5: Add sender and receiver labels  
participants\_named <- comm\_pairs %>%  
 left\_join(mc3\_nodes\_cleaned %>% select(id, sender\_label = label), by = c("sender\_id" = "id")) %>%  
 left\_join(mc3\_nodes\_cleaned %>% select(id, receiver\_label = label), by = c("receiver\_id" = "id"))  
  
  
  
# Step7: Interactive summary of top sender–receiver pairs  
participants\_named %>%  
 count(sender\_label, receiver\_label, sort = TRUE) %>%  
 datatable(  
 caption = "Top Communication Pairs (Sender → Receiver)",  
 colnames = c("Sender", "Receiver", "Message Count"),  
 options = list(pageLength = 10, autoWidth = TRUE),  
 rownames = FALSE  
 )

### **3.4b - Network Visual of Communication Participants**

This code creates an **interactive communication network graph** using visNetwork, where:

* Each **node** represents a person or entity, **node size** is based on total messages **sent** by that participant.
* Each **edge (arrow)** represents a communication sent from one participant to another, the thicker the edge, the more message sent to that particular receiver.

**Ver 1: Layout\_in\_circle**

Message Senders are arranged from most to the least number of messages sent.

library(visNetwork)  
  
# Step 1: Summarize communication edges  
comm\_edges\_vis <- participants\_named %>%  
 count(sender\_id, receiver\_id, sort = TRUE) %>%  
 rename(from = sender\_id, to = receiver\_id, value = n)  
  
# Step 2: Compute messages sent per node  
message\_counts <- comm\_edges\_vis %>%  
 group\_by(from) %>%  
 summarise(sent\_count = sum(value), .groups = "drop")  
  
# Step 3: Prepare nodes, merge with message count and add color/shape  
nodes\_vis <- mc3\_nodes\_cleaned %>%  
 filter(id %in% unique(c(comm\_edges\_vis$from, comm\_edges\_vis$to))) %>%  
 select(id, label, sub\_type) %>%  
 left\_join(message\_counts, by = c("id" = "from")) %>%  
 mutate(  
 sent\_count = replace\_na(sent\_count, 0),  
 size = rescale(sent\_count, to = c(10, 40)),  
 title = paste0(label, "<br>Sub-type: ", sub\_type,  
 ifelse(!is.na(sent\_count), paste0("<br>Sent: ", sent\_count, " messages"), "")),  
 color = case\_when(  
 sub\_type == "Person" ~ "#2ca5ff",  
 sub\_type == "Organization" ~ "#f5ee15",  
 sub\_type == "Vessel" ~ "#FB7E81",  
 sub\_type == "Group" ~ "#25e158",  
 sub\_type == "Location" ~ "#ec4bff",  
 TRUE ~ "black"  
 ),  
 shape = case\_when(  
 sub\_type == "Person" ~ "dot",  
 sub\_type == "Organization" ~ "square",  
 sub\_type == "Vessel" ~ "triangle",  
 sub\_type == "Group" ~ "star",  
 sub\_type == "Location" ~ "diamond",  
 TRUE ~ "dot"  
 ),  
 ) %>%  
 arrange(desc(size))  
  
# Step 4: Format visNetwork edges  
edges\_vis <- comm\_edges\_vis %>%  
 mutate(  
 arrows = "to",  
 width = rescale(value, to = c(1, 6)),  
 title = paste("Messages:", value)  
 )  
  
# Step 5: Define legend items  
legend\_nodes <- data.frame(  
 label = c("Person", "Organization", "Vessel", "Group", "Location"),  
 color = c("#2ca5ff", "#f5ee15", "#FB7E81", "#25e158", "#ec4bff"),  
 shape = c("dot", "square", "triangle", "star", "diamond"),  
 stringsAsFactors = FALSE  
)  
  
# Step 6: Render network with legend  
visNetwork(nodes\_vis, edges\_vis, width = "100%", height = "1000px") %>%  
 visNodes(  
 size = nodes\_vis$size  
 # color and shape are picked up from nodes\_vis columns automatically  
 ) %>%  
 visLegend(  
 addNodes = lapply(1:nrow(legend\_nodes), function(i) {  
 list(  
 label = legend\_nodes$label[i],  
 shape = legend\_nodes$shape[i],  
 color = legend\_nodes$color[i]  
 )  
 }),  
 useGroups = FALSE,  
 width = 0.15  
 ) %>%  
 visEdges(smooth = FALSE) %>%  
 visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE) %>%  
 visIgraphLayout(layout = "layout\_in\_circle") %>%  
 visPhysics(enabled = FALSE) %>%  
 visLayout(randomSeed = 1818)

**Ver 2: Layout\_on\_sphere**

From this plot, it reveals that some pairs (e.g., Miranda Jordan and Clepper Jensen) mainly communicate with each other, suggesting isolated or private channels outside the broader network.

library(visNetwork)  
  
# Step 1: Summarize communication edges  
comm\_edges\_vis <- participants\_named %>%  
 count(sender\_id, receiver\_id, sort = TRUE) %>%  
 rename(from = sender\_id, to = receiver\_id, value = n)  
  
# Step 2: Compute messages sent per person (by sender)  
message\_counts <- comm\_edges\_vis %>%  
 group\_by(from) %>%  
 summarise(sent\_count = sum(value), .groups = "drop")  
  
# Step 3: Prepare nodes with label, subtype, color, shape, and scaled size  
nodes\_vis <- mc3\_nodes\_cleaned %>%  
 filter(id %in% unique(c(comm\_edges\_vis$from, comm\_edges\_vis$to))) %>%  
 select(id, label, sub\_type) %>%  
 left\_join(message\_counts, by = c("id" = "from")) %>%  
 mutate(  
 size = if\_else(  
 sub\_type == "Person",  
 rescale(sent\_count, to = c(10, 40), na.rm = TRUE),  
 15  
 ),  
 title = paste0(label, "<br>Sub-type: ", sub\_type,  
 ifelse(!is.na(sent\_count), paste0("<br>Sent: ", sent\_count, " messages"), "")),  
 color = case\_when(  
 sub\_type == "Person" ~ "#2ca5ff",  
 sub\_type == "Organization" ~ "#f5ee15",  
 sub\_type == "Vessel" ~ "#FB7E81",  
 sub\_type == "Group" ~ "#25e158",  
 sub\_type == "Location" ~ "#ec4bff",  
 TRUE ~ "black"  
 ),  
 shape = case\_when(  
 sub\_type == "Person" ~ "dot",  
 sub\_type == "Organization" ~ "square",  
 sub\_type == "Vessel" ~ "triangle",  
 sub\_type == "Group" ~ "star",  
 sub\_type == "Location" ~ "diamond",  
 TRUE ~ "dot"  
 )  
 )  
  
# Step 4: Format edges  
edges\_vis <- comm\_edges\_vis %>%  
 mutate(  
 arrows = "to",  
 width = rescale(value, to = c(1, 6)),  
 title = paste("Messages:", value)  
 )  
  
# Step 5: Legend mapping  
legend\_nodes <- data.frame(  
 label = c("Person", "Organization", "Vessel", "Group", "Location"),  
 color = c("#2ca5ff", "#f5ee15", "#FB7E81", "#25e158", "#ec4bff"),  
 shape = c("dot", "square", "triangle", "star", "diamond"),  
 stringsAsFactors = FALSE  
)  
  
# Step 6: Render the network with layout\_on\_sphere and legend  
visNetwork(nodes\_vis, edges\_vis, width = "100%", height = "900px") %>%  
 visNodes(  
 size = nodes\_vis$size  
 # color and shape columns are automatically used  
 ) %>%  
 visLegend(  
 addNodes = lapply(1:nrow(legend\_nodes), function(i) {  
 list(  
 label = legend\_nodes$label[i],  
 shape = legend\_nodes$shape[i],  
 color = legend\_nodes$color[i]  
 )  
 }),  
 useGroups = FALSE,  
 width = 0.15  
 ) %>%  
 visEdges(smooth = FALSE) %>%  
 visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE) %>%  
 visIgraphLayout(layout = "layout\_on\_sphere") %>%  
 visPhysics(enabled = FALSE) %>%  
 visLayout(randomSeed = 1818)

## 4 - Task 1a & 1b: Daily Temporal Patterns in Communications over the Two Weeks 🎯

|  |
| --- |
| VAST Challenge Task & Question 1a and 1b |
| Clepper found that messages frequently came in at around the same time each day.   * Develop a graph-based visual analytics approach to identify any daily temporal patterns in communications. * How do these patterns shift over the two weeks of observations? |

**Objective**

1. Identify **when** communications happen most often during each day.
2. Detect **shifts in these patterns** over the 2-week period.
3. Later: Focus on a **specific entity** (e.g., Nadia Conti) and explore **who influences them**.

### Step 1: Extract & Parse Communication Event Timestamps

Extract the Communication Timestamps from mc3\_nodes\_final and filter for communication events.

# Filter for Communication events  
comm\_events <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 mutate(timestamp = ymd\_hms(timestamp)) %>%  
 filter(!is.na(timestamp)) %>%  
 mutate(  
 day = as.Date(timestamp),  
 hour = hour(timestamp)  
 )

Parse the Communication Timestamp into the format “dd/mm/yyy (ddd)” for ease of reference.

# Communication events with parsed date and time  
comm\_events <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 mutate(timestamp = ymd\_hms(timestamp)) %>%  
 filter(!is.na(timestamp)) %>%  
 mutate(  
 hour = hour(timestamp),  
 date\_label = format(timestamp, "%d/%m/%Y (%a)") # e.g., "19/03/2040 (Tue)"  
 )

### Step 2: Visualize the Communication Volume for Analysis

#### **4.1 - Bar Plot of daily communication volume over the 2 weeks period:**

# Step 1: Prepare daily message volume data  
daily\_message\_volume <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 mutate(  
 timestamp = ymd\_hms(timestamp),  
 date = as.Date(timestamp),  
 date\_label = format(timestamp, "%d/%m/%Y (%a)")  
 ) %>%  
 group\_by(date, date\_label) %>%  
 summarise(message\_count = n(), .groups = "drop") %>%  
 arrange(date)  
  
# Step 2: Compute average and total message count  
avg\_msg\_count <- mean(daily\_message\_volume$message\_count)  
total\_msg\_count <- sum(daily\_message\_volume$message\_count)  
  
# Step 3: Plot bar chart with average + total labels  
ggplot(daily\_message\_volume, aes(x = date\_label, y = message\_count)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 geom\_text(  
 aes(label = message\_count),  
 vjust = -0.3,  
 size = 2.5,  
 color = "grey40"  
 ) +  
 geom\_hline(yintercept = avg\_msg\_count, color = "red", linetype = "dashed", size = 1.2) +  
 annotate(  
 "label", x = 1, y = avg\_msg\_count + 2,  
 label = paste("Average =", round(avg\_msg\_count, 1)),  
 color = "red", fill = "grey90",  
 label.size = 0, hjust = -0.2, vjust = 3  
 ) +  
 annotate(  
 "label", x = nrow(daily\_message\_volume), y = max(daily\_message\_volume$message\_count) + 5,  
 label = paste("Total =", total\_msg\_count),  
 color = "black", fill = "lightgrey",  
 label.size = 0.3, hjust = 1.1, vjust = 1  
 ) +  
 labs(  
 title = "Daily Radio Communication Volume",  
 x = "Date",  
 y = "Message Count"  
 ) +  
 theme\_minimal(base\_size = 11) +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 plot.title = element\_text(face = "bold")  
 )

#### **4.2 - Interactive Table of daily communication volume variation(message count)**

library(DT)  
  
# Daily message volume with comparisons  
daily\_message\_volume <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 mutate(  
 timestamp = ymd\_hms(timestamp),  
 date = as.Date(timestamp),  
 date\_label = format(timestamp, "%d/%m/%Y (%a)")  
 ) %>%  
 group\_by(date, date\_label) %>%  
 summarise(message\_count = n(), .groups = "drop") %>%  
 arrange(date) %>%  
 mutate(  
 change\_from\_prev = message\_count - lag(message\_count),  
 pct\_change\_from\_prev = round((message\_count - lag(message\_count)) / lag(message\_count) \* 100, 2)  
 )  
  
datatable(  
 daily\_message\_volume %>% select(-date), # remove raw date if not needed  
 caption = "Daily Message Volume with Day-over-Day Change",  
 options = list(pageLength = 14, order = list(list(0, 'asc'))),  
 rownames = FALSE  
)

#### **4.3a - Heat Map of hourly message volume for each day over the 2 weeks period:**

This heat map is interactive and you may choose to hover on the tile to display the **date, time, and message count**

library(forcats)  
library(plotly)  
  
# Step 1: Reconstruct sender–receiver–timestamp structure  
comm\_events\_raw <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 select(event\_id = id, timestamp) %>%  
 mutate(timestamp = ymd\_hms(timestamp),  
 hour = hour(timestamp),  
 date\_label = format(timestamp, "%d/%m/%Y (%a)"))  
  
# Step 2: Get sender (sent) and receiver (received) links  
comm\_edges\_sent <- mc3\_edges\_cleaned %>%  
 filter(type == "sent") %>%  
 select(event\_id = to\_id, sender\_id = from\_id)  
  
comm\_edges\_recv <- mc3\_edges\_cleaned %>%  
 filter(type == "received") %>%  
 select(event\_id = from\_id, receiver\_id = to\_id)  
  
# Step 3: Join all together into sender–receiver–timestamp  
comm\_links <- comm\_events\_raw %>%  
 left\_join(comm\_edges\_sent, by = "event\_id") %>%  
 left\_join(comm\_edges\_recv, by = "event\_id") %>%  
 left\_join(mc3\_nodes\_cleaned %>% select(sender\_id = id, sender\_label = label), by = "sender\_id") %>%  
 left\_join(mc3\_nodes\_cleaned %>% select(receiver\_id = id, receiver\_label = label), by = "receiver\_id")  
  
# Step 4: Aggregate total messages per hour/day  
comm\_heatmap <- comm\_links %>%  
 group\_by(date\_label, hour) %>%  
 summarise(  
 count = n(),  
 top\_sender = names(sort(table(sender\_label), decreasing = TRUE))[1],  
 sender\_count = max(table(sender\_label)),  
 top\_receiver = names(sort(table(receiver\_label), decreasing = TRUE))[1],  
 receiver\_count = max(table(receiver\_label)),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 tooltip = paste0(  
 "📅 Date: ", date\_label,  
 "<br>⏰ Hour: ", sprintf("%02d:00", hour),  
 "<br>📨 Messages: ", count,  
 "<br>🔴 Top Sender: ", top\_sender, " (", sender\_count, ")",  
 "<br>🟢 Top Receiver: ", top\_receiver, " (", receiver\_count, ")"  
 )  
 )  
  
# Step 5: Static ggplot  
p <- ggplot(comm\_heatmap, aes(  
 x = hour,  
 y = fct\_rev(factor(date\_label)),  
 fill = count,  
 text = tooltip  
)) +  
 geom\_tile(color = "white") +  
 scale\_fill\_viridis\_c(option = "inferno", direction = -1, name = "Message Count") +  
 scale\_x\_continuous(  
 breaks = 0:23,  
 labels = function(x) sprintf("%02d:00", x)  
 ) +  
 labs(  
 title = "Hourly Heatmap of Radio Communications by Day",  
 x = "Hour of Day",  
 y = NULL  
 ) +  
 theme\_minimal(base\_size = 10) +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 panel.grid = element\_blank()  
 )  
  
# Step 6: Make interactive  
ggplotly(p, tooltip = "text")

We will increase the resolution to half-hour time slots.

#### **4.4b - Heat Map of half-hourly message volume for each day over the 2 weeks period:**

This heat map is interactive and you may choose to hover on the tile to display the **date, time, and message count**.

library(forcats)  
library(plotly)  
  
# Step 1: Fix sender and receiver edges  
comm\_edges\_sent <- mc3\_edges\_cleaned %>%  
 filter(type == "sent") %>%  
 select(event\_id = to\_id, sender\_id = from\_id)  
  
comm\_edges\_recv <- mc3\_edges\_cleaned %>%  
 filter(type == "received") %>%  
 select(event\_id = from\_id, receiver\_id = to\_id) # ✅ fixed receiver\_id  
  
# Step 2: Reconstruct sender–receiver–event linkage  
comm\_events\_raw <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 select(event\_id = id, timestamp) %>%  
 mutate(  
 timestamp = ymd\_hms(timestamp),  
 hour = hour(timestamp),  
 minute = minute(timestamp),  
 time\_bin = hour + ifelse(minute < 30, 0, 0.5),  
 date\_label = format(timestamp, "%d/%m/%Y (%a)"),  
 time\_label = sprintf("%02d:%02d", floor(time\_bin), ifelse(time\_bin %% 1 == 0, 0, 30))  
 )  
  
# Step 3: Join to get sender/receiver labels  
comm\_links <- comm\_events\_raw %>%  
 left\_join(comm\_edges\_sent, by = "event\_id") %>%  
 left\_join(comm\_edges\_recv, by = "event\_id") %>%  
 left\_join(mc3\_nodes\_cleaned %>% select(id, sender\_label = label), by = c("sender\_id" = "id")) %>%  
 left\_join(mc3\_nodes\_cleaned %>% select(id, receiver\_label = label), by = c("receiver\_id" = "id"))  
  
# Step 4: Aggregate by half-hour + label top actors  
comm\_heatmap <- comm\_links %>%  
 group\_by(date\_label, time\_bin, time\_label) %>%  
 summarise(  
 count = n(),  
 top\_sender = names(sort(table(sender\_label), decreasing = TRUE))[1],  
 sender\_count = max(table(sender\_label)),  
 top\_receiver = names(sort(table(receiver\_label), decreasing = TRUE))[1],  
 receiver\_count = max(table(receiver\_label)),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 tooltip = paste0(  
 "📅 Date: ", date\_label,  
 "<br>🕒 Time: ", time\_label,  
 "<br>📨 Messages: ", count,  
 "<br>🔴 Top Sender: ", top\_sender, " (", sender\_count, ")",  
 "<br>🟢 Top Receiver: ", top\_receiver, " (", receiver\_count, ")"  
 )  
 )  
  
# Step 5: ggplot  
p <- ggplot(comm\_heatmap, aes(x = time\_bin, y = fct\_rev(factor(date\_label)), fill = count, text = tooltip)) +  
 geom\_tile(color = "white") +  
 scale\_fill\_viridis\_c(  
 option = "inferno",  
 direction = -1,  
 name = "Message Count"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 labs(  
 title = "Half-Hourly Heatmap of Radio Communications by Day",  
 x = "Time of Day",  
 y = NULL  
 ) +  
 theme\_minimal(base\_size = 10) +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 panel.grid = element\_blank()  
 )  
  
# Step 6: Convert to interactive Plotly plot  
ggplotly(p, tooltip = "text")

#### **4.4c - Density plot of Daily half-hourly message volume over the 2 weeks period:**

The faceted density plot that shows the **distribution of communication events by time of day**, broken down for each day in the dataset. It helps to visually detect **temporal communication patterns**, intensity, and consistency over multiple days.

### Overview of the 2 week period

# Step 1: Preprocess communication events  
comm\_events <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 mutate(timestamp = ymd\_hms(timestamp)) %>%  
 filter(!is.na(timestamp)) %>%  
 mutate(  
 date\_label = format(timestamp, "%d/%m/%Y (%a)"),  
 hour = hour(timestamp),  
 minute = minute(timestamp),  
 time\_bin = hour + ifelse(minute < 30, 0, 0.5)  
 )  
  
# Step 2: Summarise daily medians and counts  
daily\_stats <- comm\_events %>%  
 group\_by(date\_label) %>%  
 summarise(  
 median\_time = median(time\_bin),  
 msg\_count = n(),  
 .groups = "drop"  
 )  
  
# Step 3: Plot  
ggplot(comm\_events, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = daily\_stats, aes(xintercept = median\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(  
 data = daily\_stats,  
 aes(x = 20.5, y = 0.25, label = paste("Total:", msg\_count)),  
 inherit.aes = FALSE,  
 size = 3,  
 color = "grey20",  
 hjust = 1  
 ) +  
 facet\_wrap(~ date\_label, ncol = 4) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = NULL # suppress all x-axis labels  
 ) +  
 labs(  
 title = "Daily Communication Patterns (Half-Hourly)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold"),  
 panel.grid.minor = element\_blank()  
 )

### Day 1 - 01/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "01/10/2040 (Mon)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 2 - 02/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "02/10/2040 (Tue)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 3 - 03/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "03/10/2040 (Wed)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 4 - 04/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "04/10/2040 (Thu)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 5 - 05/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "05/10/2040 (Fri)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 6 - 06/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "06/10/2040 (Sat)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 7 - 07/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "07/10/2040 (Sun)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 8 - 08/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "08/10/2040 (Mon)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 9 - 09/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "09/10/2040 (Tue)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 10 - 10/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "10/10/2040 (Wed)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 11 - 11/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "11/10/2040 (Thu)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 12 - 12/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "12/10/2040 (Fri)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 13 - 13/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "13/10/2040 (Sat)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

### Day 14 - 14/10/2040

# Filter for selected date  
daily\_data <- filter(comm\_events, date\_label == "14/10/2040 (Sun)")  
  
# Build density data to find full curve  
density\_data <- ggplot\_build(  
 ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density()  
)$data[[1]]  
  
# Identify peaks within ±10% of the maximum density  
max\_y <- max(density\_data$y)  
peak\_threshold <- 0.9 \* max\_y  
  
major\_peaks <- density\_data %>%  
 filter(y >= peak\_threshold) %>%  
 group\_by(grp = cumsum(c(1, diff(x) > 0.5))) %>% # group close bins  
 summarise(  
 peak\_time = x[which.max(y)],  
 peak\_density = max(y),  
 .groups = "drop"  
 ) %>%  
 mutate(  
 label = sprintf("Peak: %02d:%02d",  
 floor(peak\_time),  
 ifelse(peak\_time %% 1 == 0, 0, 30))  
 )  
  
# Message count  
msg\_count <- nrow(daily\_data)  
  
# Final plot  
ggplot(daily\_data, aes(x = time\_bin)) +  
 geom\_density(fill = "steelblue", alpha = 0.7) +  
 geom\_vline(data = major\_peaks, aes(xintercept = peak\_time),  
 color = "red", linetype = "dashed", linewidth = 0.6) +  
 geom\_text(data = major\_peaks,  
 aes(x = peak\_time, y = 0.23, label = label),  
 color = "red", size = 3.5, hjust = -0.1, inherit.aes = FALSE) +  
 annotate("text", x = 20.5, y = 0.25,  
 label = paste("Total messages communicated:", msg\_count),  
 hjust = 1, size = 4, color = "grey30") +  
 labs(  
 title = "Half-Hourly Communication Density (Multiple Peaks Highlighted)",  
 x = "Time of Day",  
 y = "Density"  
 ) +  
 scale\_x\_continuous(  
 breaks = seq(0, 23.5, by = 0.5),  
 labels = function(x) sprintf("%02d:%02d", floor(x), ifelse(x %% 1 == 0, 0, 30))  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 strip.text = element\_text(face = "bold")  
 )

📈 **Insights This Visualization Offers**

### Step 3: Plot Combined Hourly and Half-hourly Communication Volume

**Bar Plot of combined hourly message volume over the 2 weeks period:**

# Prepare data  
comm\_hourly <- comm\_events %>%  
 count(hour) %>%  
 mutate(  
 hour\_label = sprintf("%02d:00", hour), # Format to hh:mm  
 percent = n / sum(n)  
 )  
  
# Plot  
ggplot(comm\_hourly, aes(x = hour\_label, y = n)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 geom\_text\_repel(  
 aes(label = paste0(n, " (", percent(percent, accuracy = 1), ")")),  
 nudge\_y = 3,  
 size = 2.5,  
 direction = "y",  
 max.overlaps = Inf  
 ) +  
 labs(  
 title = "Overall Hourly Communication Volume",  
 x = "Time of Day (hh:mm)",  
 y = "Message Count"  
 ) +  
 scale\_y\_continuous(expand = expansion(mult = c(0, 0.1))) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 plot.title = element\_text(face = "bold")  
 )

**Bar Plot of combined half-hourly message volume in the 2 weeks period.**

comm\_events <- mc3\_nodes\_final %>%  
 filter(type == "Event", sub\_type == "Communication") %>%  
 mutate(timestamp = ymd\_hms(timestamp)) %>%  
 filter(!is.na(timestamp)) %>%  
 mutate(  
 hour = hour(timestamp),  
 minute = minute(timestamp),  
 time\_bin = sprintf("%02d:%02d", hour, ifelse(minute < 30, 0, 30))  
 )  
  
comm\_halfhour <- comm\_events %>%  
 count(time\_bin) %>%  
 mutate(percent = n / sum(n))  
  
ggplot(comm\_halfhour, aes(x = time\_bin, y = n)) +  
 geom\_bar(stat = "identity", fill = "steelblue") +  
 geom\_text\_repel(  
 aes(label = paste0(n, " (", percent(percent, accuracy = 1), ")")),  
 nudge\_y = 3,  
 size = 2.5,  
 direction = "y",  
 max.overlaps = Inf  
 ) +  
 labs(  
 title = "Overall Half-Hourly Communication Volume",  
 x = "Time of Day (hh:mm)",  
 y = "Message Count"  
 ) +  
 theme\_minimal() +  
 theme(  
 axis.text.x = element\_text(angle = 45, hjust = 1),  
 plot.title = element\_text(face = "bold")  
 )

|  |
| --- |
| 1a. What are the identifiable daily temporal patterns in communications? |
| 1. The daily communication volume fluctuates slightly between 34 and 49 messages, with an average of approximately 42 messages per day, highlighting a stable overall activity level. Notably, the highest volume occurs on 11th October (49 messages), immediately following the lowest volume the day before on 10th October (34 messages)—a sharp rebound that may signal a response to specific events or operational needs. Despite these fluctuations, the system maintains a consistent tempo across the two weeks. 2. The temporal analysis using both the heat map and time series plots reveals a pronounced morning-centric communication rhythm. The vast majority of radio traffic is concentrated between 9:00 AM and 11:30 AM, with the most intense peaks typically occurring between 10:00 and 11:00 AM. With reference to the Density plot of Daily half-hourly message volume, of the 14 days, we see message density peaks at 10:30 AM on 9 days, while on 3 days, it peaks at 12:30 PM. 3. For instance if we were to based in on the hourly plot, 5th October (Fri) and 11th October (Thu) both register their highest single-hour counts at 10:00 AM at 24 and 21 messages respectively. Communication activity drops off steeply after lunchtime, with more than 90% of the days showing little to no activity after 2:30 PM. This pattern suggests a highly structured daily workflow, where key decisions and coordination are front-loaded in the day. Importantly, the hourly heat map also indicates that this routine holds across both weekdays and weekends—communication volumes and peak hours remain similar, underlining the operational regularity of the group regardless of the day of week. |

|  |
| --- |
| 1b. How do these patterns shift over the two weeks of observations? |
| 1. Over the two-week period, while the timing and structure of communication peaks remain broadly consistent, there are subtle shifts in both intensity and timing. Some days, such as 3rd, 5th, 11th and 12th October, see particularly high spikes in the mid-morning, which may correspond to critical events, decision points, or heightened urgency. The sharp dip on October 8th and 13th, immediately after a period of “surge” (3rd - 7th and 9th to 12th October), points to possible responses to interruptions, lulls, or triggering incidents. Overall, although the daily messaging routine is remarkably stable, these bursts and brief lulls provide clues to changing circumstances or stress points in the operation—an analytical signal that warrants closer inspection of event logs or external triggers for those dates. 2. Another notable change in the communication pattern is observed during the weekends. In the first week, weekend communication peaks occurred earlier, typically between 10:00 AM and 11:30 AM, closely mirroring the weekday rhythm. However, in the second week, the weekend peaks shifted noticeably later, with the highest message volumes concentrated around 12:00 PM and 1:00 PM. This shift not only marks a departure from the otherwise stable early-morning communication structure but also suggests an adaptive or reactive operational schedule—potentially in response to evolving events, increased coordination needs, or changing priorities as the observation period progressed. The contrast between the two weekends is clear in the heatmap, underscoring the importance of monitoring such shifts as possible indicators of underlying changes in group behavior or external pressures. |

## 5 - Task 1c: Focus on a Particular Entity - “Nadia Conti”

|  |
| --- |
| VAST Challenge Task & Question 1c |
| Clepper found that messages frequently came in at around the same time each day.   1. Focus on a specific entity and use this information to determine who has influence over them. |

### 5.1 **-** Data Preparation for “Nadia Conti” Influence Analysis

We first extracted the relevant communication edges from the dataset, pairing “sent” and “received” communication events to form entity-to-entity links. We retained only those edges where both nodes represent real-world entities (Person, Organization, Vessel, Group, or Location), ensuring that our analysis focuses on the meaningful actors in the Oceanus network.

# Extract sent and received communication event edges  
sent\_edges <- mc3\_edges\_cleaned %>%  
 filter(type == "sent") %>%  
 select(source\_entity = from\_id, event = to\_id)  
  
received\_edges <- mc3\_edges\_cleaned %>%  
 filter(type == "received") %>%  
 select(event = from\_id, target\_entity = to\_id)  
  
# Pair sent and received to form communication edges  
paired\_edges <- sent\_edges %>%  
 inner\_join(received\_edges, by = "event") %>%  
 select(from = source\_entity, to = target\_entity)  
  
# Add unmatched sent and received edges (optional, for completeness)  
single\_sent\_edges <- sent\_edges %>%  
 select(from = source\_entity, to = event)  
single\_received\_edges <- received\_edges %>%  
 select(from = event, to = target\_entity)  
  
all\_edges <- bind\_rows(paired\_edges, single\_sent\_edges, single\_received\_edges) %>%  
 distinct()  
  
# Identify entity nodes (Person, Organization, Vessel, Group, Location)  
entity\_ids <- mc3\_nodes\_cleaned %>%  
 filter(sub\_type %in% c("Person", "Organization", "Vessel", "Group", "Location")) %>%  
 pull(id) %>% as.character()  
  
entity\_edges <- all\_edges %>%  
 filter(as.character(from) %in% entity\_ids, as.character(to) %in% entity\_ids)  
  
entity\_nodes <- mc3\_nodes\_cleaned %>%  
 filter(sub\_type %in% c("Person", "Organization", "Vessel", "Group", "Location")) %>%  
 select(id, label, sub\_type)

### 5.2 **-** Build the Global Network and Compute Centrality

Using these cleaned and filtered edges and nodes, we built a global directed graph representing the Oceanus community. We then computed key network centrality metrics for each node—PageRank, betweenness, and degree—quantifying the influence and connectivity of every entity in the overall network.

library(igraph)  
  
g <- graph\_from\_data\_frame(d = entity\_edges, vertices = entity\_nodes, directed = TRUE)  
  
# Compute centralities  
V(g)$pagerank <- page\_rank(g)$vector  
V(g)$betweenness <- betweenness(g)  
V(g)$degree <- degree(g)

### 5.3 **-** Extract “Nadia Conti” Ego Network (2-hop Neighbourhood)

Focusing on “Nadia Conti”, we identified her node and extracted her two-step ego network, capturing both direct and indirect connections within the broader network. This local subgraph reveals Nadia’s immediate sphere of influence and the key players connected to her.

nadia\_label <- "Nadia Conti"  
target\_index <- which(V(g)$label == nadia\_label)  
  
ego\_graph <- make\_ego\_graph(g, order = 2, nodes = target\_index, mode = "all")[[1]]

### 5.4 **-** Visualize Nadia Conti’s Ego Network (Interactive)

We visualized Nadia’s ego network using node size, shape, and color to represent centrality and entity type. We also summarized centrality metrics in clear tables, ranking all ego network members by PageRank, Betweenness, and Degree. This allows for direct identification of the most influential, best-connected, and most strategic actors in Nadia Conti’s communication environment.

nodes\_df <- data.frame(  
 id = V(ego\_graph)$name,  
 label = V(ego\_graph)$label,  
 group = V(ego\_graph)$sub\_type,  
 title = paste0("<b>", V(ego\_graph)$label, "</b><br>",  
 "Degree: ", round(V(ego\_graph)$degree, 2), "<br>",  
 "Betweenness: ", round(V(ego\_graph)$betweenness, 2), "<br>",  
 "PageRank: ", round(V(ego\_graph)$pagerank, 4)),  
 shape = ifelse(V(ego\_graph)$sub\_type == "Person", "dot",  
 ifelse(V(ego\_graph)$sub\_type == "Organization", "square",  
 ifelse(V(ego\_graph)$sub\_type == "Vessel", "triangle",  
 ifelse(V(ego\_graph)$sub\_type == "Group", "star", "diamond")))),  
 value = V(ego\_graph)$pagerank \* 30 + 5  
)  
  
edges\_df <- as\_data\_frame(ego\_graph, what = "edges") %>%  
 rename(from = from, to = to)  
  
library(visNetwork)  
visNetwork(nodes\_df, edges\_df, width = "100%", height = "700px") %>%  
 visNodes(scaling = list(min = 5, max = 30)) %>%  
 visEdges(  
 arrows = list(to = list(enabled = TRUE, scaleFactor = 0.3)),  
 color = list(color = "gray")  
 ) %>%  
 visOptions(  
 highlightNearest = TRUE,  
 nodesIdSelection = TRUE,  
 manipulation = FALSE  
 ) %>%  
 visInteraction(  
 dragNodes = FALSE,  
 dragView = FALSE,  
 zoomView = FALSE  
 ) %>%  
 visLegend() %>%  
 visLayout(randomSeed = 1818)

|  |
| --- |
| Global and Ego-Network Structure |
| The overview network visualization reveals that **Nadia Conti** is centrally embedded in the Oceanus communication web, maintaining direct and indirect connections with major actors such as Neptune (Vessel), V. Miesel Shipping (Organization), Elise (Person), and others. The use of color and shape coding in the network allows for quick identification of the different types of entities in Nadia’s influence neighborhood. |

### 5.5 **-** Centrality Tables for Nadia’s Ego Network

On both the global and Nadia-focused ego networks, we computed standard network centrality metrics for all nodes:

* **PageRank** (overall influence),
* **Betweenness** (information brokerage/intermediary role), and
* **Degree** (number of direct connections).

These measures quantify the importance and structural roles of each entity relative to Nadia and the broader community.

# PageRank table  
pagerank\_df <- data.frame(  
 label = V(ego\_graph)$label,  
 sub\_type = V(ego\_graph)$sub\_type,  
 pagerank = round(V(ego\_graph)$pagerank, 4)  
) %>% arrange(desc(pagerank))  
  
# Betweenness table  
betweenness\_df <- data.frame(  
 label = V(ego\_graph)$label,  
 sub\_type = V(ego\_graph)$sub\_type,  
 betweenness = round(V(ego\_graph)$betweenness, 2)  
) %>% arrange(desc(betweenness))  
  
# Degree table  
degree\_df <- data.frame(  
 label = V(ego\_graph)$label,  
 sub\_type = V(ego\_graph)$sub\_type,  
 degree = V(ego\_graph)$degree  
) %>% arrange(desc(degree))

knitr::kable(pagerank\_df, caption = "PageRank Centrality (Nadia's Ego Network)")

knitr::kable(betweenness\_df, caption = "Betweenness Centrality (Nadia's Ego Network)")

knitr::kable(degree\_df, caption = "Degree Centrality (Nadia's Ego Network)")

|  |
| --- |
| Centrality Metrics and Direct & Indirect Influences |
| By calculating centrality metrics within Nadia’s two-hop ego network, we observe that the most influential nodes in her environment—by PageRank, betweenness, and degree—are Neptune, V. Miesel Shipping, and Elise. Nadia herself consistently ranks among the top nodes by these measures, highlighting her role as both an influencer and an information bridge. Entities such as Neptune and V. Miesel Shipping, which also score highly in centrality, exert considerable influence over Nadia’s information flow and access to other parts of the network.  Degree centrality analysis shows Nadia maintains multiple direct connections, particularly with other highly active nodes, ensuring she is closely linked to key hubs in the network. Betweenness centrality further reveals that Nadia is not only well-connected but also acts as an important intermediary, facilitating communication between otherwise distant parts of the network. PageRank confirms that her immediate environment is composed of actors with significant structural power, increasing the likelihood that Nadia is both influenced by, and exerts influence upon, the most pivotal players in Oceanus. |

#### **5.5.1 - PageRank for Nadia Conti**

library(igraph)  
library(visNetwork)  
  
# -- Build the global network g as in your earlier code (using your entity\_nodes/entity\_edges) --  
  
g <- graph\_from\_data\_frame(  
 d = entity\_edges,   
 vertices = entity\_nodes,   
 directed = TRUE  
)  
  
# -- Get Nadia's index in g --  
nadia\_label <- "Nadia Conti"  
target\_index <- which(V(g)$label == nadia\_label)  
  
# -- Extract Nadia's 1-hop ego network (all direct neighbors) --  
ego\_1 <- make\_ego\_graph(g, order = 1, nodes = target\_index, mode = "all")[[1]]  
  
  
# 1. Compute PageRank for the ego network  
V(ego\_1)$pagerank <- page\_rank(ego\_1)$vector  
  
# 2. Prepare node data frame with your consistent color scheme  
nodes\_df\_pagerank <- data.frame(  
 id = V(ego\_1)$name,  
 label = V(ego\_1)$label,  
 group = V(ego\_1)$sub\_type,  
 title = paste0("<b>", V(ego\_1)$label, "</b><br>PageRank: ", round(V(ego\_1)$pagerank, 4)),  
 shape = ifelse(V(ego\_1)$sub\_type == "Person", "dot",  
 ifelse(V(ego\_1)$sub\_type == "Organization", "square",  
 ifelse(V(ego\_1)$sub\_type == "Vessel", "triangle",  
 ifelse(V(ego\_1)$sub\_type == "Group", "star", "diamond")))),  
 color = case\_when(  
 V(ego\_1)$sub\_type == "Person" ~ "#2ca5ff",  
 V(ego\_1)$sub\_type == "Organization" ~ "#f5ee15",  
 V(ego\_1)$sub\_type == "Vessel" ~ "#FB7E81",  
 V(ego\_1)$sub\_type == "Group" ~ "#25e158",  
 V(ego\_1)$sub\_type == "Location" ~ "#ec4bff",  
 TRUE ~ "black"  
 ),  
 value = V(ego\_1)$pagerank \* 30 + 5  
)  
  
# 3. Prepare edges  
edges\_df <- as\_data\_frame(ego\_1, what = "edges") %>%  
 rename(from = from, to = to)  
  
# 4. Plot with visNetwork  
visNetwork(nodes\_df\_pagerank, edges\_df, width = "100%", height = "400px") %>%  
 visNodes(  
 scaling = list(min = 5, max = 30),  
 color = list(background = nodes\_df\_pagerank$color, border = "black"),  
 shape = nodes\_df\_pagerank$shape  
 ) %>%  
 visEdges(arrows = list(to = list(enabled = TRUE, scaleFactor = 0.3)), color = list(color = "gray")) %>%  
 visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE, manipulation = FALSE) %>%  
 visInteraction(dragNodes = FALSE, dragView = FALSE, zoomView = FALSE) %>%  
 visLegend() %>%  
 visLayout(randomSeed = 1818)

#### **5.5.2 - Betweenness for Nadia Conti**

# 1. Compute Betweenness for the ego network  
V(ego\_1)$betweenness <- betweenness(ego\_1, directed = TRUE)  
  
# 2. Prepare node data frame  
nodes\_df\_betweenness <- data.frame(  
 id = V(ego\_1)$name,  
 label = V(ego\_1)$label,  
 group = V(ego\_1)$sub\_type,  
 title = paste0("<b>", V(ego\_1)$label, "</b><br>Betweenness: ", round(V(ego\_1)$betweenness, 2)),  
 shape = ifelse(V(ego\_1)$sub\_type == "Person", "dot",  
 ifelse(V(ego\_1)$sub\_type == "Organization", "square",  
 ifelse(V(ego\_1)$sub\_type == "Vessel", "triangle",  
 ifelse(V(ego\_1)$sub\_type == "Group", "star", "diamond")))),  
 color = case\_when(  
 V(ego\_1)$sub\_type == "Person" ~ "#2ca5ff",  
 V(ego\_1)$sub\_type == "Organization" ~ "#f5ee15",  
 V(ego\_1)$sub\_type == "Vessel" ~ "#FB7E81",  
 V(ego\_1)$sub\_type == "Group" ~ "#25e158",  
 V(ego\_1)$sub\_type == "Location" ~ "#ec4bff",  
 TRUE ~ "black"  
 ),  
 value = V(ego\_1)$betweenness \* 2 + 5  
)  
  
# 3. Edges (same as before)  
# edges\_df already prepared  
  
# 4. Plot  
visNetwork(nodes\_df\_betweenness, edges\_df, width = "100%", height = "400px") %>%  
 visNodes(  
 scaling = list(min = 5, max = 30),  
 color = list(background = nodes\_df\_betweenness$color, border = "black"),  
 shape = nodes\_df\_betweenness$shape  
 ) %>%  
 visEdges(arrows = list(to = list(enabled = TRUE, scaleFactor = 0.3)), color = list(color = "gray")) %>%  
 visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE, manipulation = FALSE) %>%  
 visInteraction(dragNodes = FALSE, dragView = FALSE, zoomView = FALSE) %>%  
 visLegend() %>%  
 visLayout(randomSeed = 1818)

#### **5.5.3 - Degree for Nadia Conti**

# 1. Compute Degree for the ego network  
V(ego\_1)$degree <- degree(ego\_1, mode = "all")  
  
# 2. Prepare node data frame  
nodes\_df\_degree <- data.frame(  
 id = V(ego\_1)$name,  
 label = V(ego\_1)$label,  
 group = V(ego\_1)$sub\_type,  
 title = paste0("<b>", V(ego\_1)$label, "</b><br>Degree: ", round(V(ego\_1)$degree, 2)),  
 shape = ifelse(V(ego\_1)$sub\_type == "Person", "dot",  
 ifelse(V(ego\_1)$sub\_type == "Organization", "square",  
 ifelse(V(ego\_1)$sub\_type == "Vessel", "triangle",  
 ifelse(V(ego\_1)$sub\_type == "Group", "star", "diamond")))),  
 color = case\_when(  
 V(ego\_1)$sub\_type == "Person" ~ "#2ca5ff",  
 V(ego\_1)$sub\_type == "Organization" ~ "#f5ee15",  
 V(ego\_1)$sub\_type == "Vessel" ~ "#FB7E81",  
 V(ego\_1)$sub\_type == "Group" ~ "#25e158",  
 V(ego\_1)$sub\_type == "Location" ~ "#ec4bff",  
 TRUE ~ "black"  
 ),  
 value = V(ego\_1)$degree \* 5 + 5  
)  
  
# 3. Edges (same as before)  
# edges\_df already prepared  
  
# 4. Plot  
visNetwork(nodes\_df\_degree, edges\_df, width = "100%", height = "400px") %>%  
 visNodes(  
 scaling = list(min = 5, max = 30),  
 color = list(background = nodes\_df\_degree$color, border = "black"),  
 shape = nodes\_df\_degree$shape  
 ) %>%  
 visEdges(arrows = list(to = list(enabled = TRUE, scaleFactor = 0.3)), color = list(color = "gray")) %>%  
 visOptions(highlightNearest = TRUE, nodesIdSelection = TRUE, manipulation = FALSE) %>%  
 visInteraction(dragNodes = FALSE, dragView = FALSE, zoomView = FALSE) %>%  
 visLegend() %>%  
 visLayout(randomSeed = 1818)

|  |
| --- |
| 1c. With a focus on “Nadia Conti”, the visuals above could determine who has influence over this person. |
| 1. Degree centrality reveals that Nadia Conti is well-connected within her local network, with a degree of 17. However, she is not the most connected node; vessels such as Mako (37), Reef Guardian (27), and Remora (21), as well as organizations like Oceanus City Council (28) and V. Miesel Shipping (19), have even higher degrees. This indicates that while Nadia is an important hub, her sphere of direct interaction is embedded within a dense mesh of other highly connected entities. 2. Several other individuals (e.g., Davis with 16, Boss with 13, Mrs. Money with 12) and vessels (e.g., Neptune with 19, Sentinel with 16) also play significant roles in Nadia’s network. The presence of organizations (e.g., Green Guardians, Sailor Shifts Team), multiple vessels, and key persons shows that **Nadia’s environment is both diverse and robust**. 3. **Direct Connections**  * These direct connections are clearly shown as nodes that have edges (arrows) going into or out of Nadia Conti’s node in the network diagrams. Nadia Conti directly connects to several core entities across different types:   + **People:** Elise, Liam Thorne, Davis, Rodriguez   + **Organization:** V. Miesel Shipping, Oceanus City Council, Sailor Shifts Team   + **Vessel:** Neptune, Marlin, Remora, Sentinel   + **Location:** Haacklee Harbor * **Interpretation:** The PageRank, Betweenness, and Degree centrality plots all consistently show **Nadia Conti as a major hub**, with a large node size reflecting her high centrality. Her immediate network includes influential vessels (Neptune, Remora), organizations (V. Miesel Shipping, Oceanus City Council), and several persons (Elise, Davis, Rodriguez). * Nadia’s position suggests she is a **key connector and influencer** but is herself surrounded by even larger hubs, particularly among vessels and organizations. Her ability to influence—and be influenced—is amplified by these connections, as these high-degree entities are likely sources and conduits of critical information and operational coordination. This structure points to a tightly interwoven community, where central actors such as Mako, Oceanus City Council, and V. Miesel Shipping may exert the most substantial influence over Nadia’s access to information, resources, and strategic decisions. |