Graph Analysis project: Diseases Spreading Process in the US Air Transportation Network With and Without Covid-19 Restrictions

Théo Condette, Maria Bracque Vendrell

2024-03-03

1. Create Dataset and Graph Object

First, let's import the data:

```
#Théo

#flights_2021 <- read.csv2("C:/Users/conde/Desktop/Code Cours/TSE/Master 2/Graph Analysis/Project/fligh
#flights_2019 <- read.csv2("C:/Users/conde/Desktop/Code Cours/TSE/Master 2/Graph Analysis/Project/fligh


#Maria

flights_2021 <- read.csv2("C:/Users/maria/Documents/00 TSE/M2/Graph Analysis/Project/flights_2021.csv",s
flights_2019 <- read.csv2("C:/Users/maria/Documents/00 TSE/M2/Graph Analysis/Project/flights_2019.csv",s
```

Before transforming the data into a graph, let's prepare the flight data for 2019 by calculating the total number of flights between each pair of cities. We decided to do a weighted graph where the nodes are the cities, the edges are the annual routes, and the weights are the number of times a route was made during the year. We also decided to work by looking at the flights between cities instead of airports to avoid having too many nodes since some cities have several airports.

```
flights_2019 <- flights_2019%>%
                    select(ORIGIN_CITY_NAME,DEST_CITY_NAME)

# Let's compute the relation weight between two airports:
  # High weight means high number of flights from A to B
  # Low weight means low number of flights from A to B

flights_2019$Weight <- 1

final_flights_2019 <- flights_2019%>%
  group_by(ORIGIN_CITY_NAME,DEST_CITY_NAME)%>%
  summarise(Total_flights = sum(Weight))

relations_2019 <- data.frame(From =final_flights_2019$ORIGIN_CITY_NAME,
                             To =final_flights_2019$DEST_CITY_NAME,Total_flights = final_flights_2019$T

actors_2019 <- data.frame(final_flights_2019%>%
```

```
  group_by(ORIGIN_CITY_NAME)%>%
  summarize(Total_flight_airport =sum(Total_flights)))
```

Let's check if all edges are from and to existing nodes:

```
all_nodes_2019 <- unique(c(relations_2019$From, relations_2019$To))
existing_nodes_2019 <- unique(actors_2019$ORIGIN_CITY_NAME)

missing_nodes_2019 <- setdiff(all_nodes_2019, existing_nodes_2019)

if (length(missing_nodes_2019) == 0) {
  print("All edges are from existing nodes.")
} else {
  print("Missing nodes:")
  print(missing_nodes_2019)
}
```

```
## [1] "All edges are from existing nodes."
```

Let's do the same for 2021:

```
flights_2021 <- flights_2021%>%
                    select(ORIGIN_CITY_NAME,DEST_CITY_NAME)

# Let's compute the relation weight between two airports:
  # High weight means high number of flights from A to B
  # Low weight means low number of flights from A to B

flights_2021$Weight <- 1

final_flights_2021 <- flights_2021%>%
  group_by(ORIGIN_CITY_NAME,DEST_CITY_NAME)%>%
  summarise(Total_flights = sum(Weight))

relations_2021 <- data.frame(From =final_flights_2021$ORIGIN_CITY_NAME,To =final_flights_2021$DEST_CITY

actors_2021 <- data.frame(final_flights_2021%>%
  group_by(ORIGIN_CITY_NAME)%>%
  summarize(Total_flight_airport =sum(Total_flights)))


# We want to keep only the same actors as in 2019 so we can compare the results when doing the ABM:

  # Keep only the nodes existing in 2019:
actors_2021 <- actors_2021[which(actors_2021$ORIGIN_CITY_NAME %in% actors_2019$ORIGIN_CITY_NAME),]
```

Let's check if all edges are from and to existing nodes:

```
all_nodes_2021 <- unique(c(relations_2021$From, relations_2021$To))
existing_nodes_2021 <- unique(actors_2021$ORIGIN_CITY_NAME)

missing_nodes_2021 <- setdiff(all_nodes_2021, existing_nodes_2021)
```

```r
if (length(missing_nodes_2021) == 0) {
  print("All edges are from existing nodes.")
} else {
  print("Missing nodes:")
  print(missing_nodes_2021)
}
```

```
## [1] "Missing nodes:"
##  [1] "Alamosa, CO"           "Bishop, CA"            "Cold Bay, AK"
##  [4] "Decatur, IL"           "Dickinson, ND"         "Dodge City, KS"
##  [7] "Fort Dodge, IA"        "Fort Leonard Wood, MO" "Johnstown, PA"
## [10] "Mason City, IA"        "Pullman, WA"           "Riverton/Lander, WY"
## [13] "Sheridan, WY"          "Victoria, TX"          "Walla Walla, WA"
## [16] "Wenatchee, WA"         "Wilmington, DE"        "Yakima, WA"
```

We see that some edges are from or go to some missing nodes. This is because we eliminated the nodes from 2021 that were not in 2019. So, let's eliminate these edges:

```r
relations_2021 <- relations_2021[-which(!relations_2021$To%in%actors_2021$ORIGIN_CITY_NAME),]
relations_2021 <- relations_2021[-which(!relations_2021$From%in%actors_2021$ORIGIN_CITY_NAME),]
```

Let's check that all these nodes have been eliminated:

```r
all_nodes_2021 <- unique(c(relations_2021$From, relations_2021$To))
existing_nodes_2021 <- unique(actors_2021$ORIGIN_CITY_NAME)

missing_nodes_2021 <- setdiff(all_nodes_2021, existing_nodes_2021)

if (length(missing_nodes_2021) == 0) {
  print("All edges are from existing nodes.")
} else {
  print("Missing nodes:")
  print(missing_nodes_2021)
}
```

```
## [1] "All edges are from existing nodes."
```

Perfect, now we are good to go and we can create the igraph objects:

```r
g_flights_2019 <- graph_from_data_frame(relations_2019, directed = TRUE,vertices=actors_2019)
E(g_flights_2019)$weight <- relations_2019$Total_flights

g_flights_2021 <- graph_from_data_frame(relations_2021, directed = TRUE,vertices=actors_2021)
E(g_flights_2021)$weight <- relations_2021$Total_flights
```

2. Graph Description

First, let's do the summary information for the graph objects:

```r
print("Graph 2019")
```

```
## [1] "Graph 2019"
```

```r
summary(g_flights_2019)
```

```
## IGRAPH 8d4fc1c DNW- 352 6039 --
## + attr: name (v/c), Total_flight_airport (v/n), Total_flights (e/n),
## | weight (e/n)
```

```r
print("Graph 2021")
```

```
## [1] "Graph 2021"
```

```r
summary(g_flights_2021)
```

```
## IGRAPH 8d50fa5 DNW- 347 6266 --
## + attr: name (v/c), Total_flight_airport (v/n), Total_flights (e/n),
## | weight (e/n)
```

The graph with the data from 2019 has 352 nodes whereas the graph from 2021 has 347 nodes (cities) (which is normal since we only kept the cities in common). For the 2019 graph, there are 6039 directed edges (routes). And for the 2021 graph, there are 6266 directed edges (routes).

Now, let us have a look at the edge density of the networks. We have to keep in mind that this density is in terms of cities and not the number of flights since the function *edge_density* does not consider weights.

```r
print("Graph 2019")
```

```
## [1] "Graph 2019"
```

```r
edge_density(g_flights_2019)
```

```
## [1] 0.04887821
```

```r
print("Graph 2021")
```

```
## [1] "Graph 2021"
```

```r
edge_density(g_flights_2021)
```

```
## [1] 0.0521897
```

The edge density for the flight network in 2019 is approximately 0.048, while for the network in 2021, it's around 0.052.

We can see that both densities are pretty similar. This could indicate a similar network structure in terms of connectivity. However, we have to keep in mind that this metric is not very relevant in this case since we have a weighted graph and the function *edge_density* does not consider weights.

We can now look at the diameter of the flight networks.

```r
print("Graph 2019")
```

```
## [1] "Graph 2019"
```

```r
diameter(g_flights_2019)
```

```
## [1] 4153
```

```r
print("Graph 2021")
```

```
## [1] "Graph 2021"
```

```r
diameter(g_flights_2021)
```

```
## [1] 3158
```

The diameter of a network represents the longest shortest path between any two vertices, indicating the maximum number of flights required to travel between the furthest cities in the network. The decrease in diameter from 2019 to 2021 suggests an improvement in overall connectivity and accessibility of cities through air travel, as the longest shortest path between cities decreased over time.

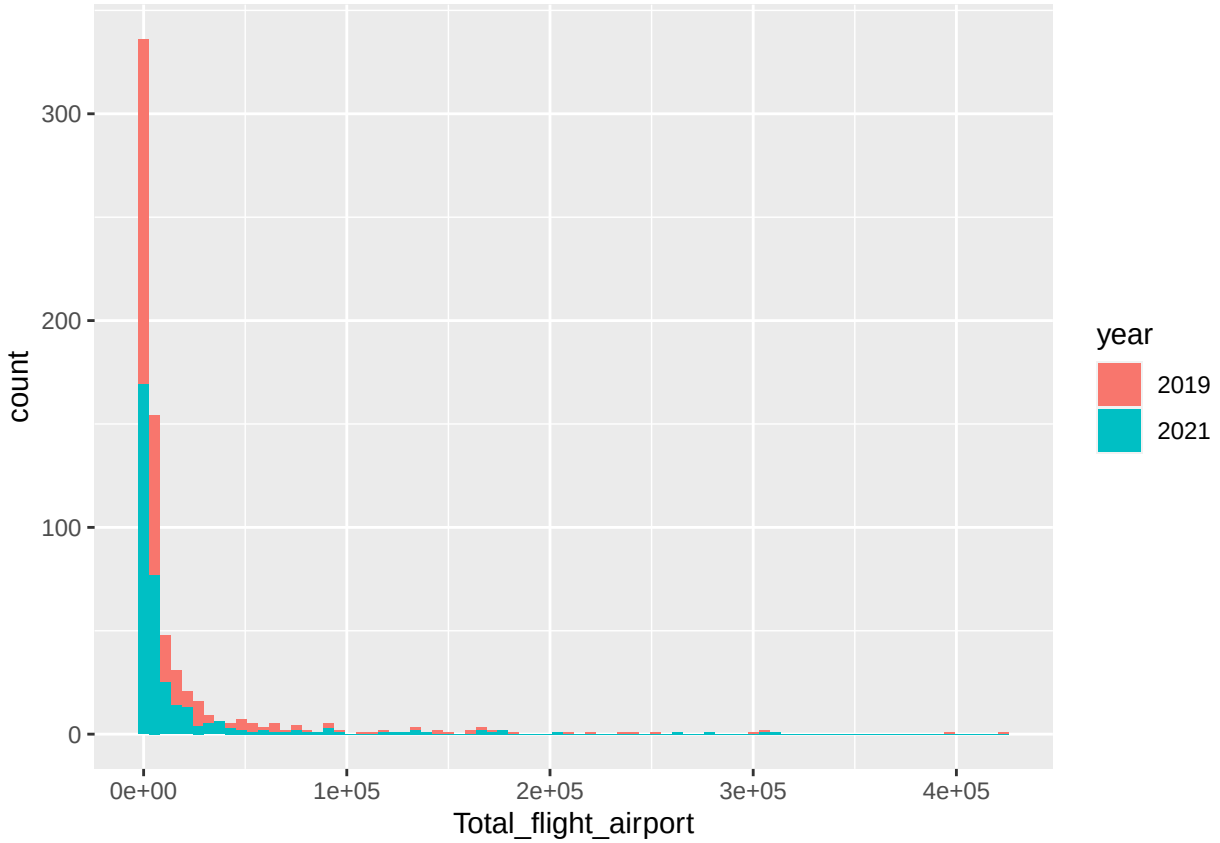
Now, let's look at the degree distribution:

```r
degree_2019<-data.frame(degree=igraph::degree(g_flights_2019))
degree_2019$year ="2019"

degree_2021<-data.frame(degree=igraph::degree(g_flights_2021))
degree_2021$year ="2021"

degree_db<-rbind(degree_2019,degree_2021)

ggplot(degree_db)+
  geom_histogram(aes(x=degree,fill=year), bins = 80)+
  xlab("Number of destination of a city")+
  ylim(c(0,160))+
  xlim(c(0,350))
```

There are fewer destinations for airports in 2021 compared to 2019, as indicated by the degree distribution histograms. This observation could suggest several possibilities, such as changes in flight routes, reductions in airline services, or alterations in travel patterns due to COVID-19 restrictions.

But what about the number of flights? Let's look at it now:

```
actors_2019$year <-"2019"
actors_2021$year <-"2021"
Comparison_df <- rbind(actors_2019,actors_2021)
ggplot(Comparison_df)+
  geom_histogram(aes(x=Total_flight_airport,fill=year), bins = 80)
```



The histogram comparing the total number of flights per airport between 2019 and 2021 confirms the observation of reduced connectivity in 2021 compared to 2019.

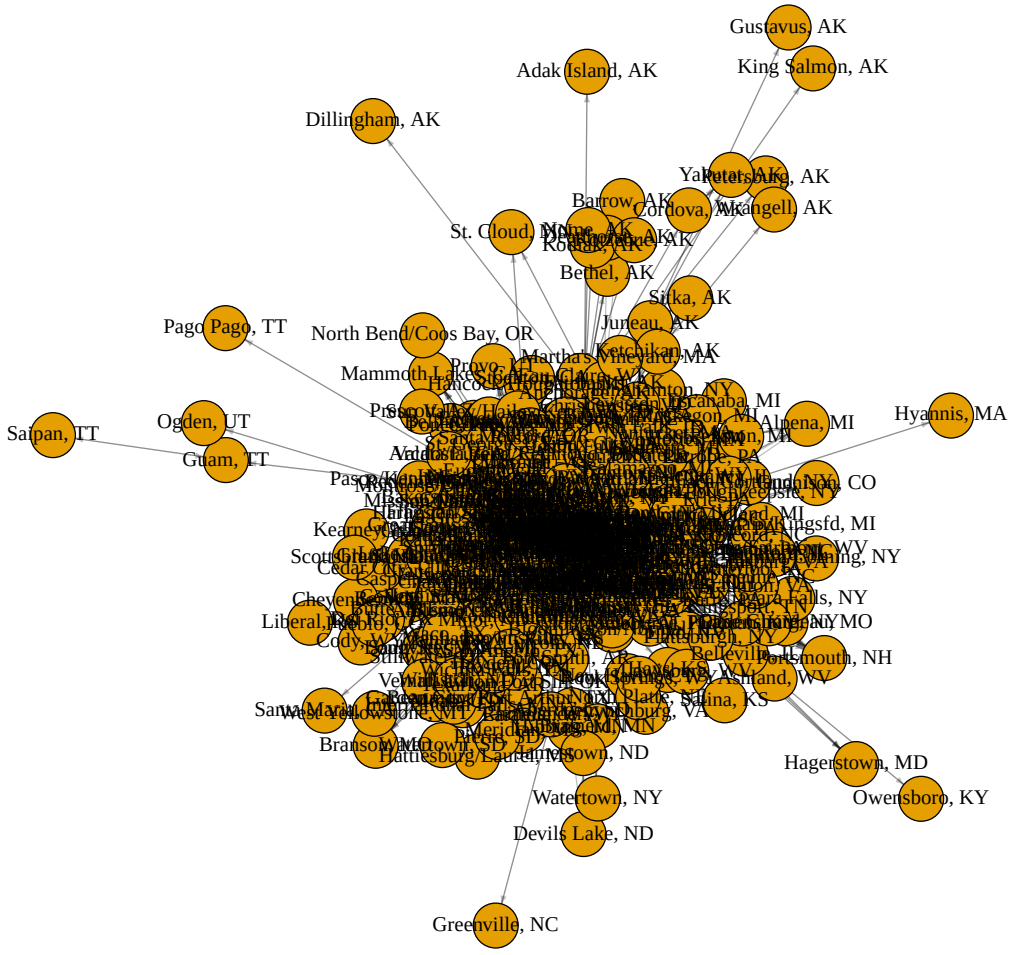
3. Graph Representation

3.1 2019 Graph

First, let's plot the graph for 2019.

Let's start by plotting the full graph without taking the weights into account, to have a first overview:

```
plot(g_flights_2019,
     vertex.size = 10,
     vertex.label.cex=1,
     vertex.label.dist=0,
     vertex.label.color = "black",
     edge.color = rgb(0,0,0,.25),
     edge.arrow.size=0.3)
```

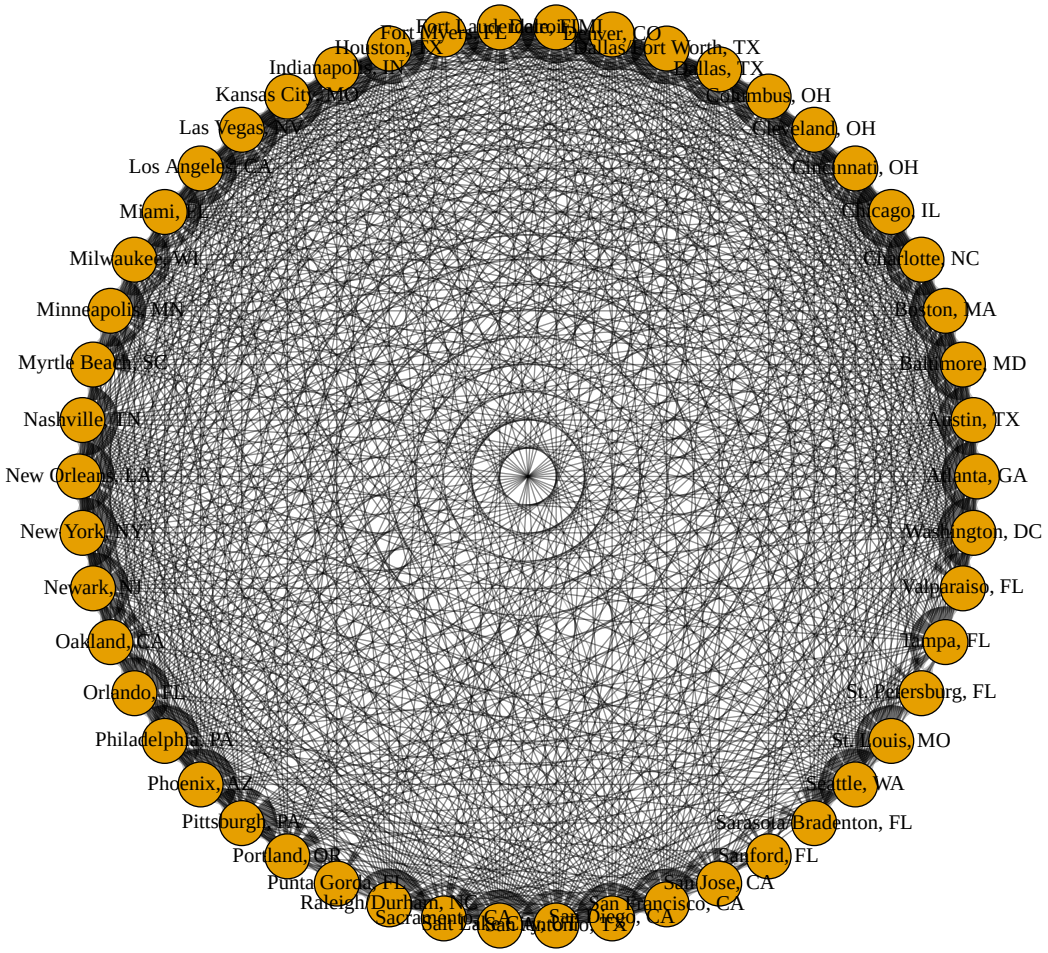


We can see that it is a very busy graph due to the important amount of nodes. We can however see that some cities are only connected to one city (for example, Saipan TT is only connected to Guam TT) whereas some are connected to a lot of cities (for example New York NY or Denver CO).

We will thus plot only the top 50 busiest airports by looking at the degrees:

```
top_50_cities_2019 <- degree_2019 %>%
  arrange(desc(degree)) %>%
  head(50)

top_50_cities_2019 <- top_50_cities_2019 %>%
  rownames_to_column(var = "name")

subgraph_2019 <- induced_subgraph(g_flights_2019, which(V(g_flights_2019)$name %in% top_50_cities_2019$n
```

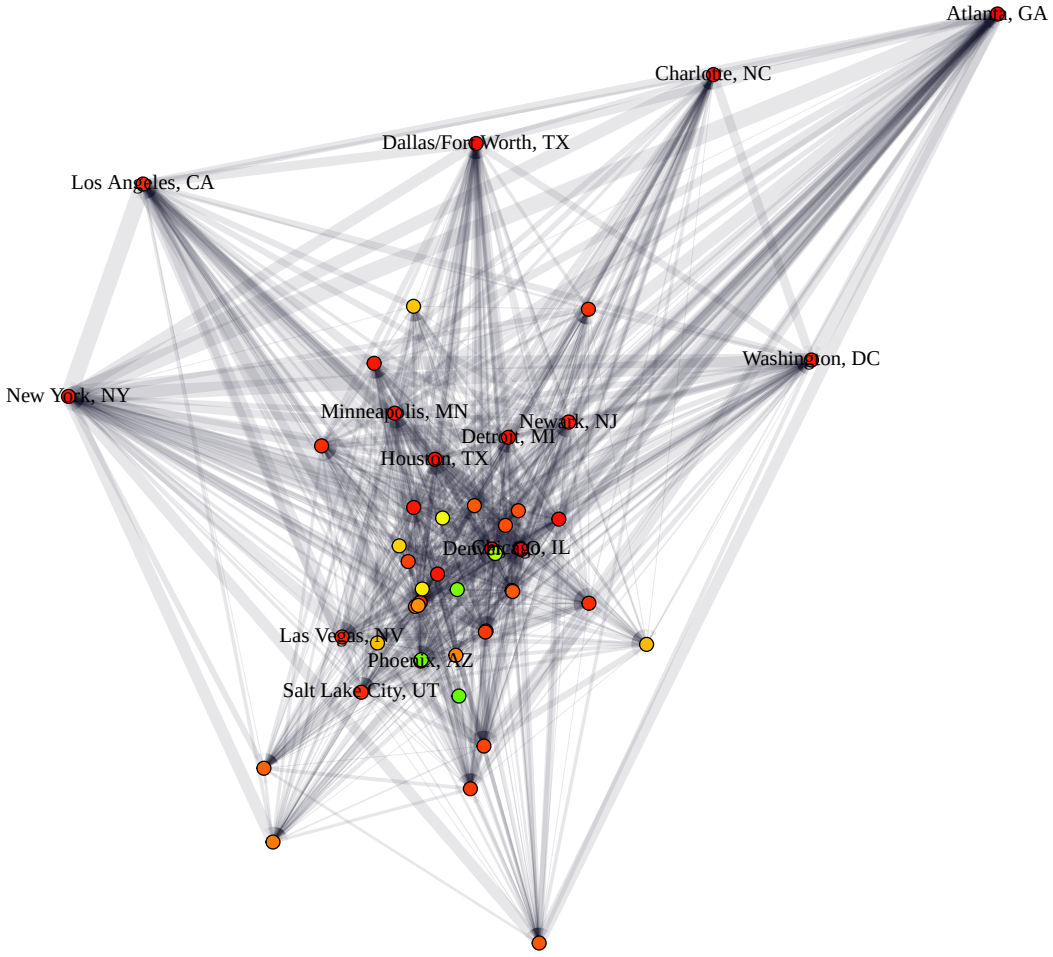
```
plot(subgraph_2019,
     layout = layout.circle,
     vertex.size = 10,
     vertex.label.cex=1,
     vertex.label.dist=0,
     vertex.label.color = "black",
     edge.color = rgb(0,0,0,.25),
     edge.arrow.size=0.5)
```



This plot shows us that among the top 50 cities, the graph is very dense.

Let's now add the weights to the graph:

```r
# Normalize edge weights
maximum_number_flight_2019 <- max(E(subgraph_2019)$weight)
E(subgraph_2019)$weight <- E(subgraph_2019)$weight / maximum_number_flight_2019

set.seed(123)
subgraph_2019$layout <- layout.kamada.kawai(subgraph_2019)

# We only want to plot the names of the top 10 to make the plot readable
cities_2019_to_show <- (top_50_cities_2019 %>% head(15))$name
sub_name_2019 <- names(V(subgraph_2019))
sub_name_2019 [!names(V(subgraph_2019))%in%cities_2019_to_show]<-NA

# Let's color the nodes according to their degree
V(subgraph_2019)$degree_in <-igraph::degree(subgraph_2019, mode="in")
color_palette <- colorRampPalette(c("green", "yellow", "red"))(max(V(subgraph_2019)$degree_in) + 1)


plot(subgraph_2019,
     vertex.label = sub_name_2019,
     vertex.label.color = "black",
     vertex.size = 3,
     vertex.color = color_palette[V(subgraph_2019)$degree_in + 1],
     edge.arrow.size = 0.3,
     vertex.label.cex = 1,
     edge.width = (E(subgraph_2019)$weight*10)*2,
     edge.color = rgb(0.1, 0.1, 0.2, 0.05))
```

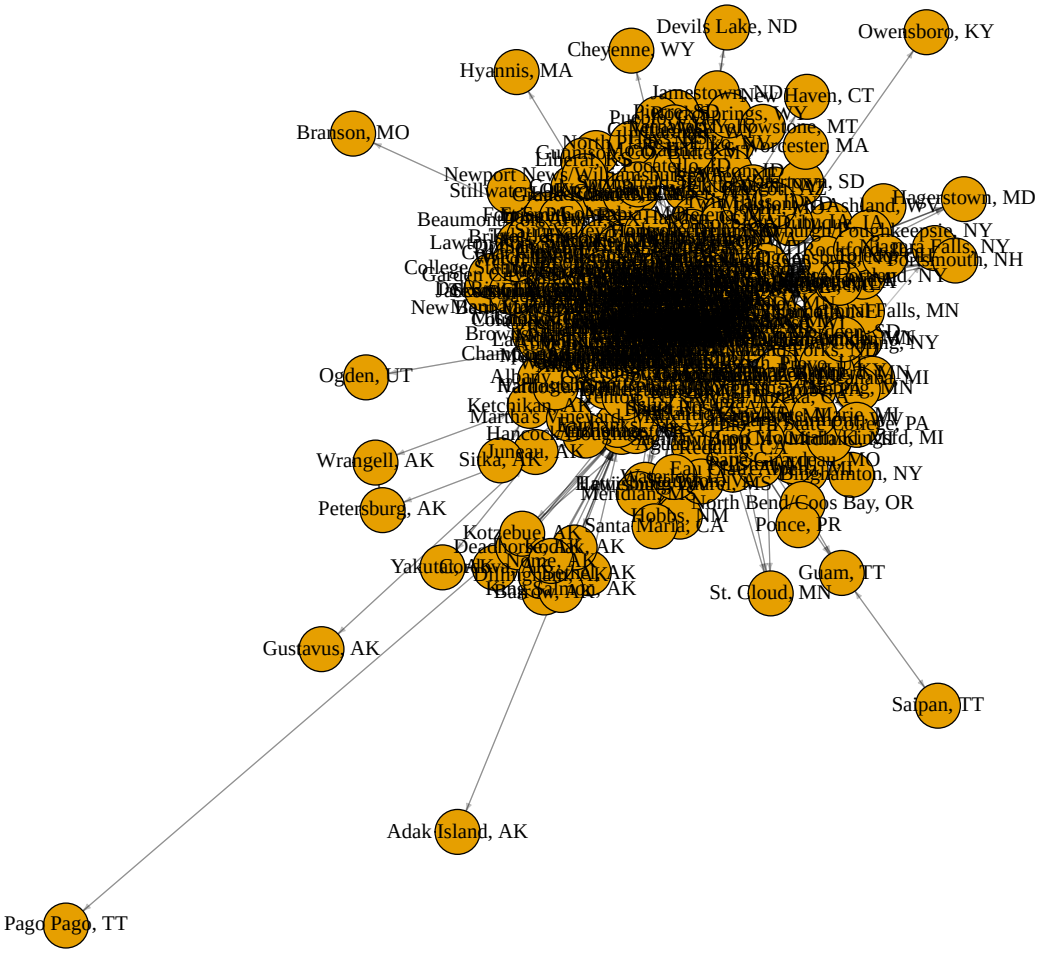


3.2 2021 Graph

Now, let's do the same for the graph for 2021.

Let's start by plotting the full graph without taking the weights into account, to have a first overview:

```
plot(g_flights_2021,
     vertex.size = 10,
     vertex.label.cex = 1,
     vertex.label.dist = 0,
     vertex.label.color = "black",
     edge.color = rgb(0, 0, 0, 0.25),
     edge.arrow.size = 0.3)
```

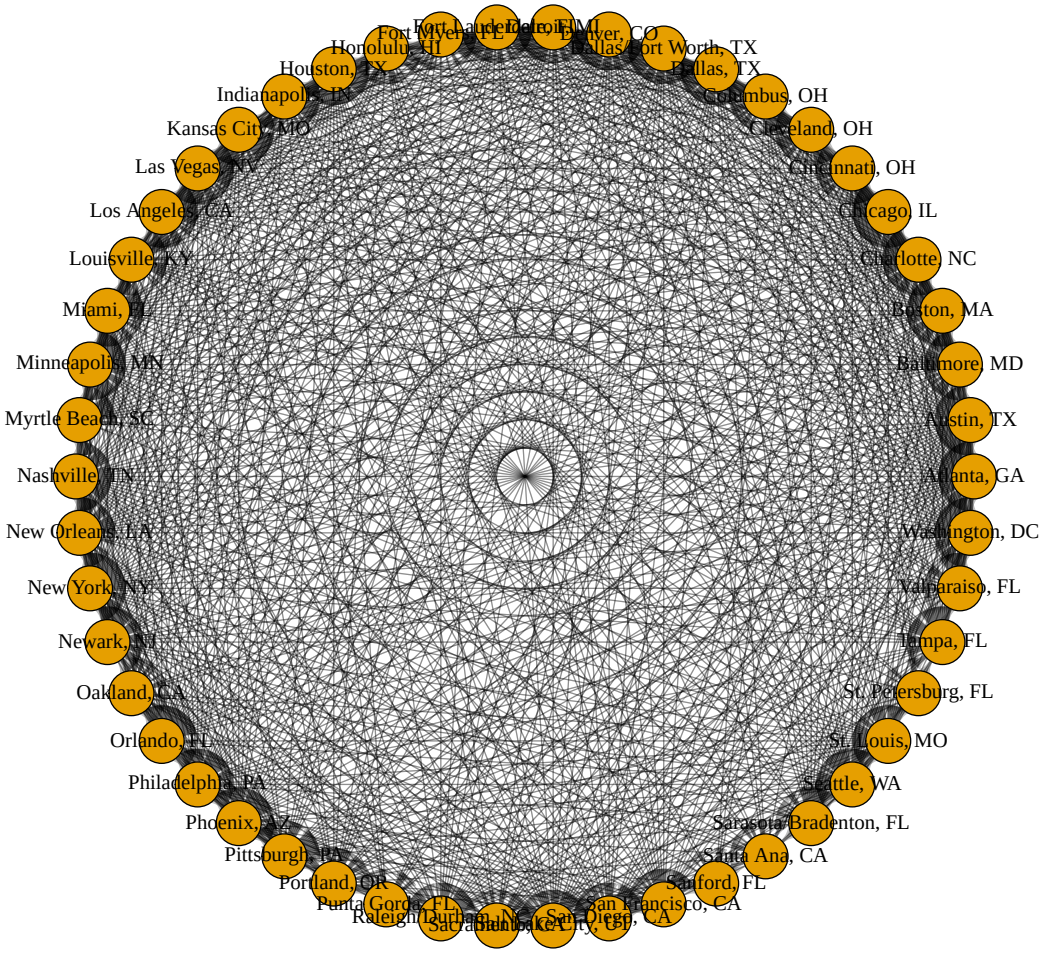


We can see that it is a very busy graph do to the important amount of nodes. Just as for 2019, we can see that some cities are only connected to one city (for example, Saipan TT is only connected to Guam TT) whereas some are connected to a lot of cities (for example New York NY or Denver CO).

We will thus plot only the top 50 busiest airports looking at the degree:

```
top_50_cities_2021 <- degree_2021 %>%
  arrange(desc(degree)) %>%
  head(50)

top_50_cities_2021 <- top_50_cities_2021 %>%
  rownames_to_column(var = "name")

subgraph_2021 <- induced_subgraph(g_flights_2021, which(V(g_flights_2021)$name %in% top_50_cities_2021$
```

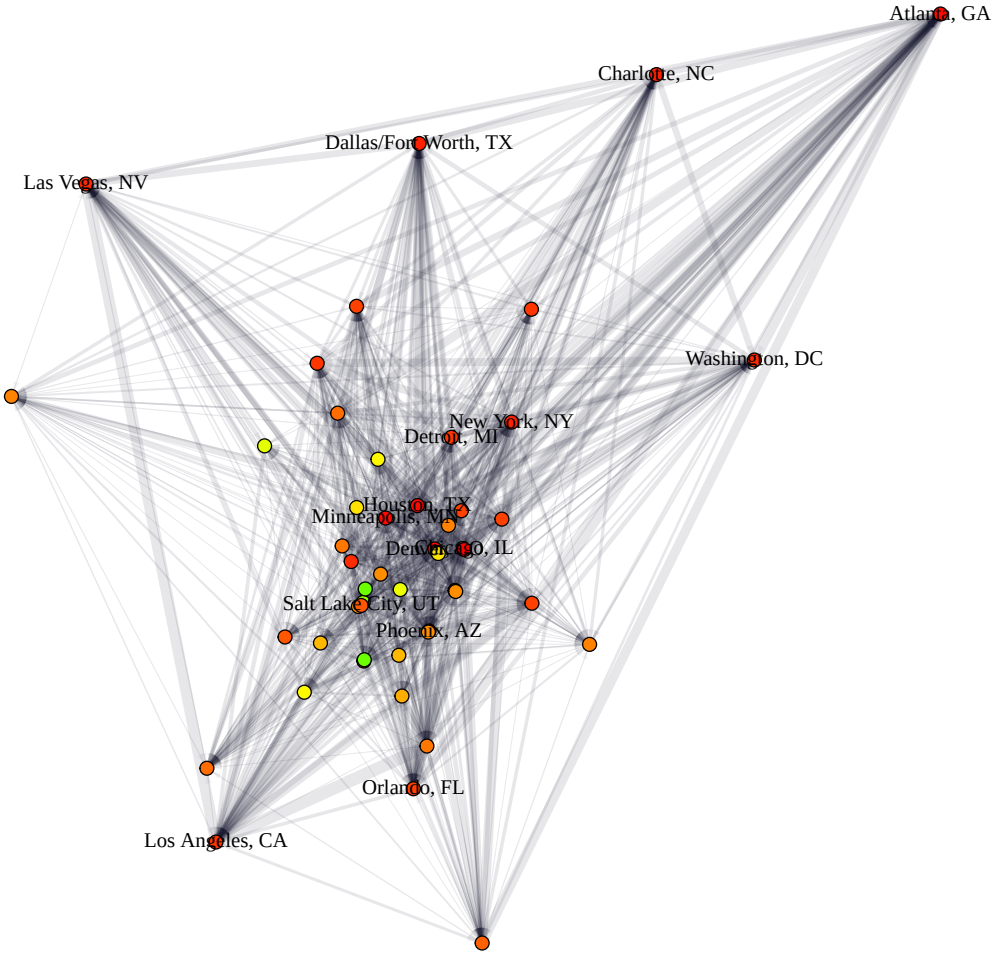
```
plot(subgraph_2021,
     layout = layout.circle,
     vertex.size = 10,
     vertex.label.cex=1,
     vertex.label.dist=0,
     vertex.label.color = "black",
     edge.color = rgb(0,0,0,.25),
     edge.arrow.size=0.5)
```



This plot shows us that among the top 50 cities the graph is very dense, like the one for 2019.

Let's now add the weights on the graph:

```r
# Normalize edge weights by the max of 2019 to compare the graphs
E(subgraph_2021)$weight <- E(subgraph_2021)$weight / maximum_number_flight_2019

# We want this graph to have the same layout as the 2019 one
subgraph_2021$layout <- subgraph_2019$layout

# We only want to plot the names of the top 10 to make the plot readable
cities_2021_to_show <- (top_50_cities_2021 %>% head(15))$name
sub_name_2021 <- names(V(subgraph_2021))
sub_name_2021 [!names(V(subgraph_2021))%in%cities_2021_to_show]<-NA

# Let's color the nodes according to their degree
V(subgraph_2021)$degree_in <-igraph::degree(subgraph_2021, mode="in")
color_palette <- colorRampPalette(c("green", "yellow", "red"))(max(V(subgraph_2021)$degree_in) + 1)


plot(subgraph_2021,
     vertex.label = sub_name_2021,
     vertex.label.color = "black",
     vertex.size = 3,
     vertex.color = color_palette[V(subgraph_2021)$degree_in + 1],
     edge.arrow.size = 0.3,
     vertex.label.cex = 1,
     edge.width = (E(subgraph_2021)$weight*10)*2,
     edge.color = rgb(0.1, 0.1, 0.2, 0.05))
```



In this graph, we can see that compared to the previous one, the number of flights is lower because the width of the arrows is smaller. Also, there are changes in the degree of some airports.

4. Centrality Measure

To identify the nodes that are at higher risk of being infected, let's calculate the weighted degree centrality.

```
# for 2019
centrality_2019 <- strength(g_flights_2019, V(g_flights_2019), mode = "out", weights = E(g_flights_2019)

  # To print the 10 cities with the higher Strength centrality measure
sorted_cities_2019 <- sort(centrality_2019, decreasing = TRUE)
top_10_cities_2019 <- names(sorted_cities_2019[1:10])
```

```r
top_10_strength_2019 <- sorted_cities_2019[1:10]


# for 2021
centrality_2021 <- strength(g_flights_2021, V(g_flights_2021), mode = "out", weights = E(g_flights_2021

  # To print the 10 cities with the higher Strength centrality measure
sorted_cities_2021 <- sort(centrality_2021, decreasing = TRUE)
top_10_cities_2021 <- names(sorted_cities_2021[1:10])
top_10_strength_2021 <- sorted_cities_2021[1:10]


print("2019")
```

```
## [1] "2019"
```

```r
for (i in 1:10) {
  print(paste("City:", top_10_cities_2019[i], "- Strength Centrality:", top_10_strength_2019[i]))}
```

```
## [1] "City: Chicago, IL - Strength Centrality: 423369"
## [1] "City: Atlanta, GA - Strength Centrality: 395009"
## [1] "City: Dallas/Fort Worth, TX - Strength Centrality: 304344"
## [1] "City: New York, NY - Strength Centrality: 298810"
## [1] "City: Denver, CO - Strength Centrality: 252026"
## [1] "City: Houston, TX - Strength Centrality: 240103"
## [1] "City: Charlotte, NC - Strength Centrality: 235496"
## [1] "City: Los Angeles, CA - Strength Centrality: 219952"
## [1] "City: Washington, DC - Strength Centrality: 208611"
## [1] "City: Phoenix, AZ - Strength Centrality: 181042"
```

```r
print("2021")
```

```
## [1] "2021"
```

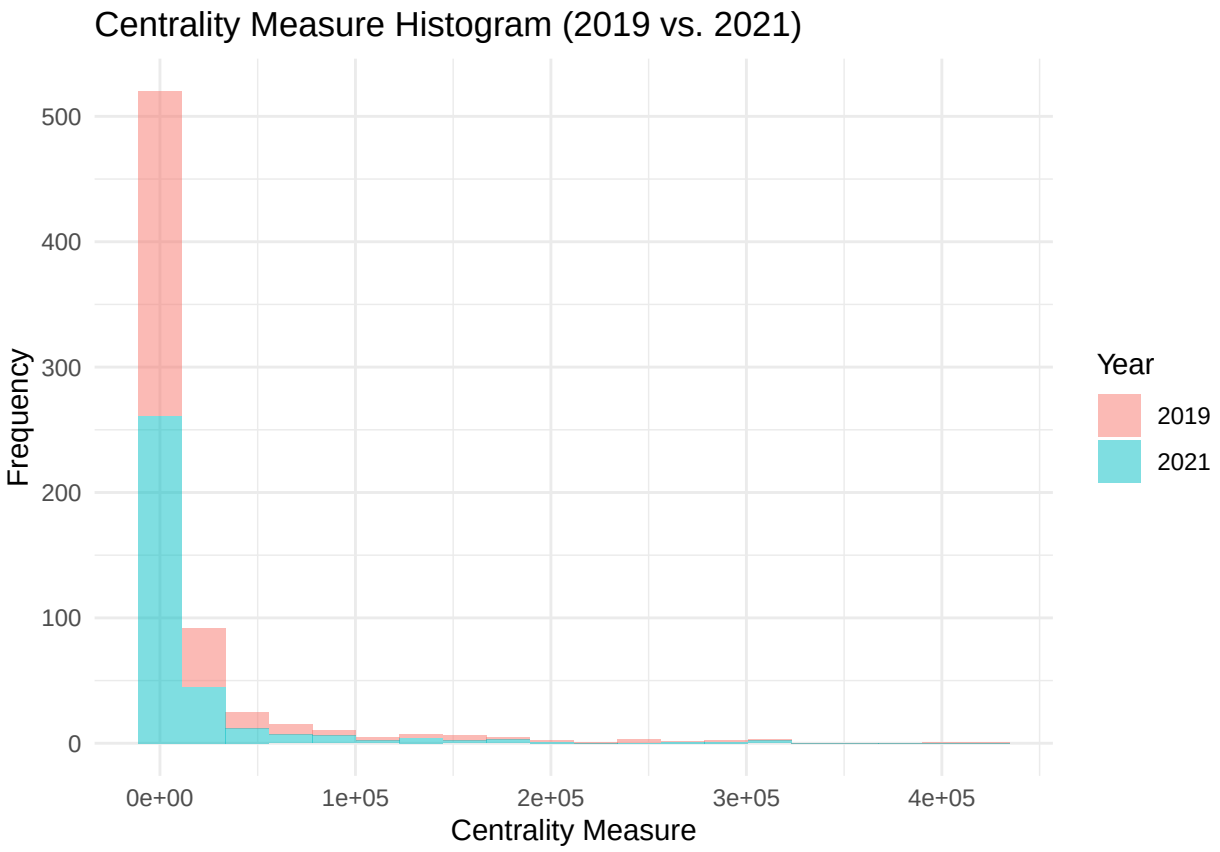
```r
for (i in 1:10) {
  print(paste("City:", top_10_cities_2021[i], "- Strength Centrality:", top_10_strength_2021[i]))}
```

```
## [1] "City: Atlanta, GA - Strength Centrality: 313299"
## [1] "City: Chicago, IL - Strength Centrality: 303349"
## [1] "City: Dallas/Fort Worth, TX - Strength Centrality: 280418"
## [1] "City: Denver, CO - Strength Centrality: 258532"
## [1] "City: Charlotte, NC - Strength Centrality: 201895"
## [1] "City: Houston, TX - Strength Centrality: 176854"
## [1] "City: Los Angeles, CA - Strength Centrality: 174305"
## [1] "City: New York, NY - Strength Centrality: 170057"
## [1] "City: Phoenix, AZ - Strength Centrality: 166639"
## [1] "City: Seattle, WA - Strength Centrality: 164833"
```

We can see that even if the order of the top 10 is different between years, the cities are the same (except for Washington, DC which has been replaced by Seattle in 2021).

Let's visualize these measures on a graph:

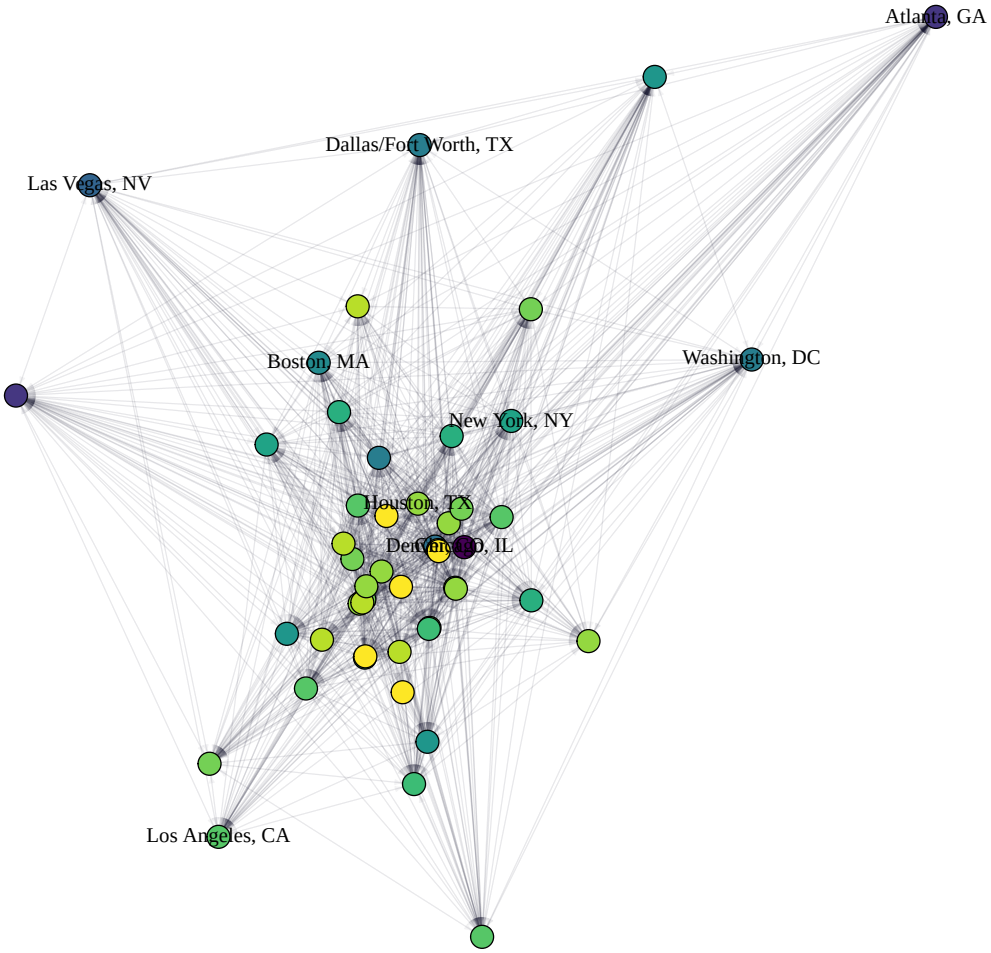
```r
centrality_2019 <- data.frame(Centrality = centrality_2019, Year = "2019")
centrality_2021 <- data.frame(Centrality = centrality_2021, Year = "2021")

centrality_db <- rbind(centrality_2019, centrality_2021)

ggplot(centrality_db) +
  geom_histogram(aes(x = Centrality, fill = Year), bins = 20, alpha = 0.5) +
  labs(title = "Centrality Measure Histogram (2019 vs. 2021)",
       x = "Centrality Measure",
       y = "Frequency") +
  theme_minimal()
```



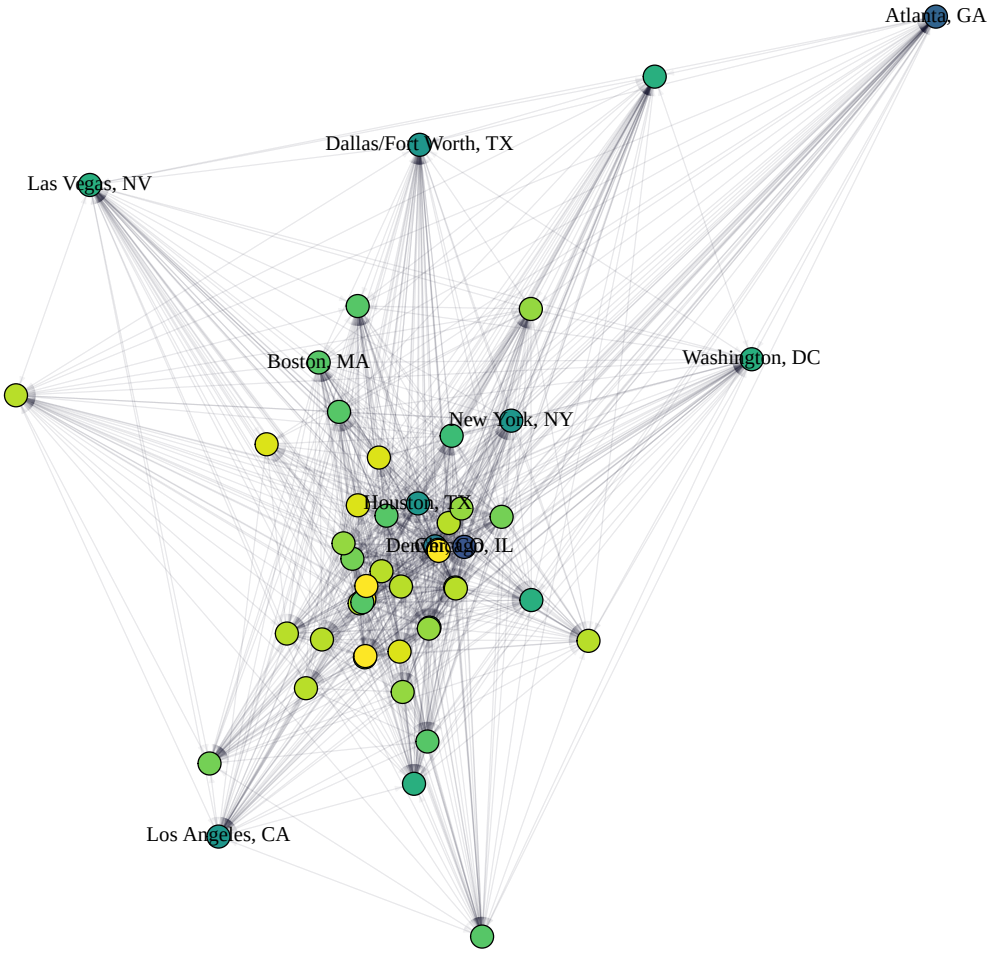
Knowing this, let's plot the graph taking into account this centrality measure.

```r
sub_centrality_2019 <- strength(subgraph_2019, V(subgraph_2019), mode = "in", weights = E(subgraph_2019)
sub_centrality_2021 <- strength(subgraph_2021, V(subgraph_2021), mode = "in", weights = E(subgraph_2021)

# We normalize it with 2019 to have the same color scale to compare
V(subgraph_2019)$centrality <- sub_centrality_2019/max(sub_centrality_2019)
V(subgraph_2021)$centrality <- sub_centrality_2021/max(sub_centrality_2019)
  # Define breakpoints :
break_points <- seq(0,1,0.05)


# We only want to plot the names of the top 10 to make the plot readable
cities_to_show_central <- names(sort(sub_centrality_2019, decreasing = TRUE)[1:10])
```

```
sub_name_central <- names(V(subgraph_2021))
sub_name_central [!names(V(subgraph_2021))%in%cities_to_show_central]<-NA


plot(subgraph_2019,
     vertex.size = 5,
     vertex.label = sub_name_central,
     vertex.label.cex = 1,
     vertex.label.dist = 0,
     vertex.label.color = "black",
     vertex.color = rev(viridis::viridis(20))[cut(V(subgraph_2019)$centrality ,breaks=break_points)],
     edge.color = rgb(0.1, 0.1, 0.2, 0.05),
     edge.arrow.size = 0.3)
```



```
plot(subgraph_2021,
     vertex.size = 5,
     vertex.label = sub_name_central,
     vertex.label.cex = 1,
     vertex.label.dist = 0,
     vertex.label.color = "black",
     vertex.color = rev(viridis::viridis(20))[cut(V(subgraph_2021)$centrality ,breaks=break_points)],
     edge.color = rgb(0.1, 0.1, 0.2, 0.05),
     edge.arrow.size = 0.3)
```



In this graph, the nodes' color go in a range between yellow and purple, with purple being the highest centrality and yellow the lowest one. We can see that some nodes are not from the same color anymore between the two years. This is due to the restrictions! However, Los Angeles airport has more flights in 2021 than in 2019. This looks to be the only one.

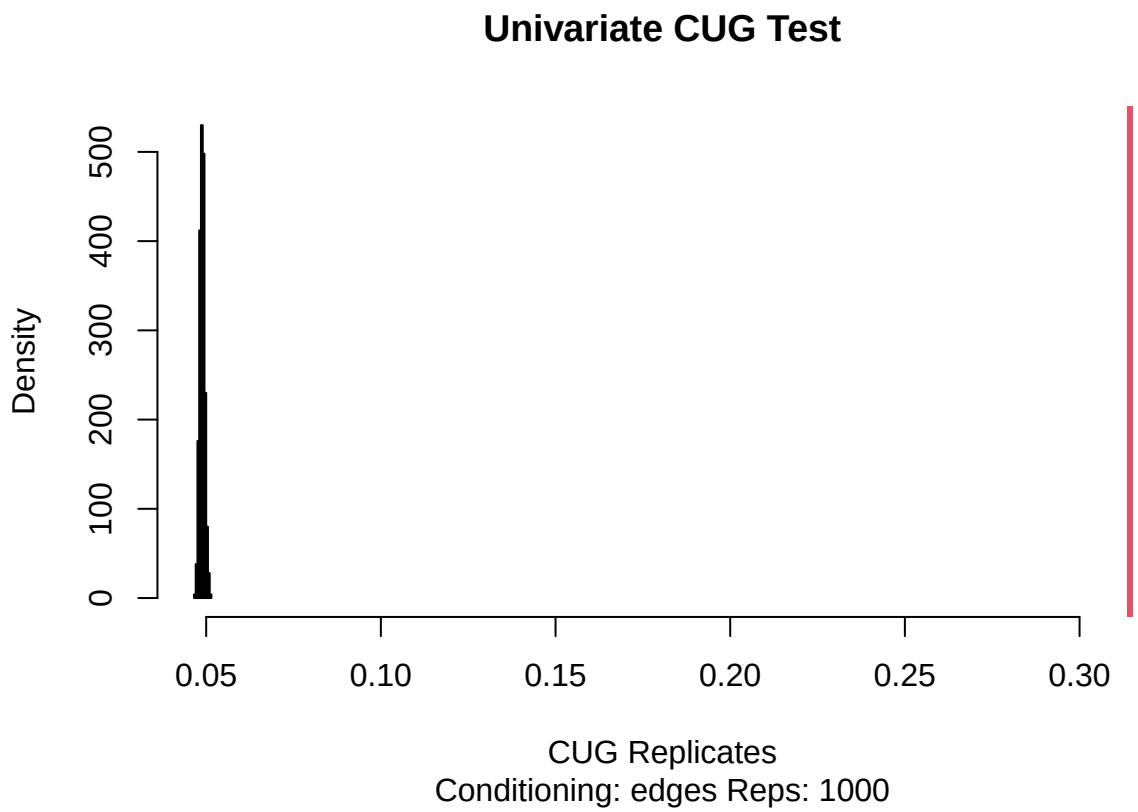
5. CUG Test

We are going to use the sna package to perform a CUG test.

```
flights_2019_adj <- as.matrix(as_adjacency_matrix(g_flights_2019))
cugtest_2019_trans <- cug.test(flights_2019_adj, FUN=gtrans, mode="digraph", cmode="edges", reps = 1000)
cugtest_2019_trans
```

```
##
## Univariate Conditional Uniform Graph Test
##
## Conditioning Method: edges
## Graph Type: digraph
## Diagonal Used: FALSE
## Replications: 1000
##
## Observed Value: 0.3144243
## Pr(X>=Obs): 0
## Pr(X<=Obs): 1
```

```
plot(cugtest_2019_trans)
```

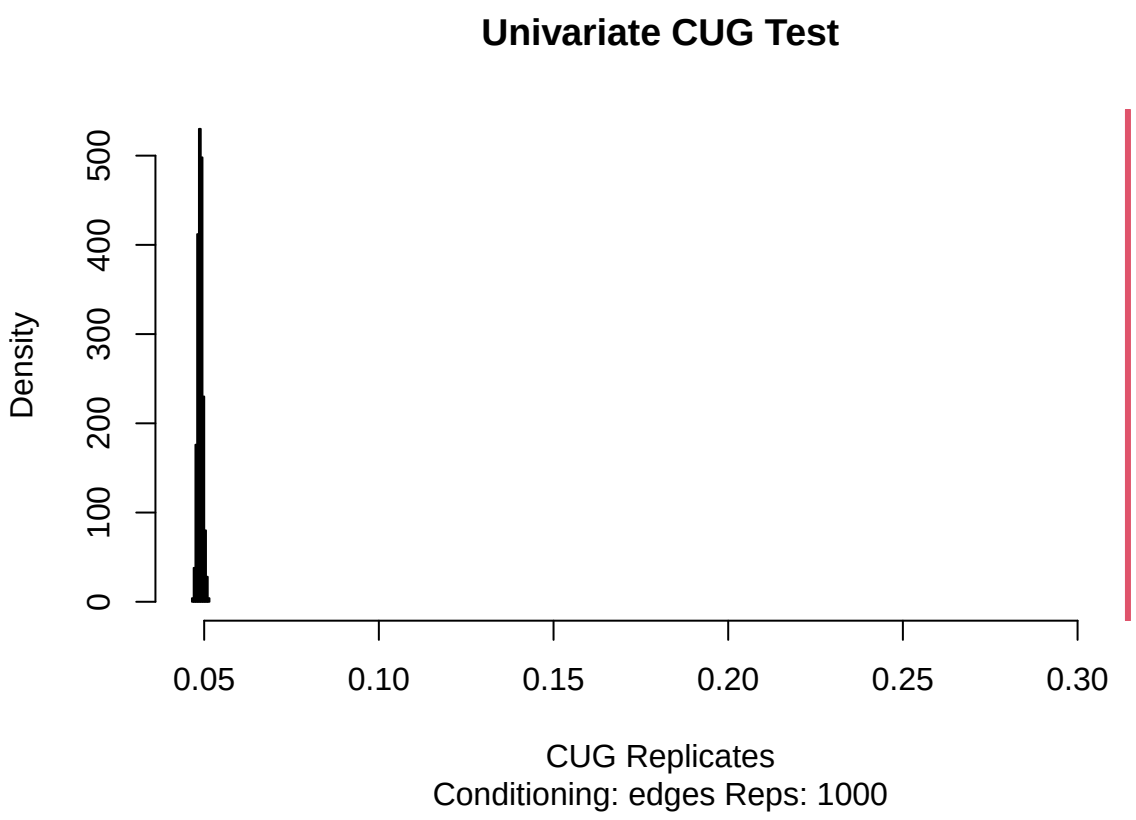


We can conclude that the observed transitivity is significantly higher (under the significance threshold of 0.05) than what would be expected at random.

```r
flights_2021_adj <- as.matrix(as_adjacency_matrix(g_flights_2021))
cugtest_2021_trans <- cug.test(flights_2021_adj, FUN=gtrans, mode="digraph", cmode="edges", reps = 1000)
cugtest_2021_trans
```

```
##
## Univariate Conditional Uniform Graph Test
##
## Conditioning Method: edges
## Graph Type: digraph
## Diagonal Used: FALSE
## Replications: 1000
##
## Observed Value: 0.3186355
## Pr(X>=Obs): 0
## Pr(X<=Obs): 1
```

```r
plot(cugtest_2019_trans)
```



Same, we can conclude that the observed transitivity is significantly higher (under the significance threshold of 0.05) than what would be expected at random.

Now, let's move to the next part of the project which is the ABM model.

6. Agent-based Model

In this section, we want to model the spread of a virus among the two networks and see if the virus spreads quicker in 2019 than in 2021. This will allow us to see if the flight restrictions during the pandemic of COVID-19 were effective (we will not be evaluating other restrictions such as shielding measures or vaccination restrictions). We are going to vary the transmission rate and see if it has an impact on the transmissions.

First, let's define a function to simulate the whole spreading process.

```
simulate_disease_spread_abm <- function(g,g2019, beta, gamma, n_seeds, max_steps) {

    # Initialization of storing results:

  infected_prop <- rep(NA,max_steps) # Gives the proportion of infected people in the US for each day.
  step_infection_actual <- numeric(vcount(g)) # Gives the proportion of infected people by city in the
  history <- list() # History is a list that store the step_infection_actual for each day from day 1 to

  set.seed(33)
  seeds <- sample(1:vcount(g),n_seeds) # Select randomly cities that have the disease inside of their p

  step_infection_actual[seeds] <- runif(n = seeds,0,0.2) # Select randomly the level of infection of th

  step_infection_next <- step_infection_actual # Initialize step_infection_next which helps to handle t
  history[[1]] <- step_infection_actual # Initialize history.
  infected_prop[1] <- sum(step_infection_actual )/vcount(g) # Initialize the step infection total rate.

  adjacency_normalized <- as.matrix(igraph::get.adjacency(g,attr = "weight"))/365 # Get the adjacency o
  adjacency_normalized_2019 <- as.matrix(igraph::get.adjacency(g2019,attr = "weight"))/365# Get the adj
  maximum_daily_2019 <- max(apply(as.matrix(igraph::get.adjacency(g2019,attr = "weight"))/365,2,sum))



  for(i in colnames(adjacency_normalized)){

      adjacency_normalized[,i]<-adjacency_normalized[,i]/maximum_daily_2019 # We normalize by the 2019
  }

  for (step in 2:max_steps) {
    for (agent in 1:vcount(g)) {
      # We stop the time, each agent is affected by the actual situation of its neighbor. The change is

        id_neighbors <- which(V(g)%in%neighbors(g, agent)) # neigbors id

        step_infection_next[agent] <- step_infection_actual[agent] + (1-step_infection_actual[agent])*t
        # The infection rate of an agent at the next step is the actual infection rate plus the non aff
        # The idea is that the affected proportion of the city does not change, what changes is the one
    }

        # We get the infected proportion of the total US population
    infected_prop[step] <- sum(step_infection_actual)/vcount(g)
    # We get the history step.
    history[[step]] <- step_infection_actual
    # When all agent have updated their situation, we can go forward for a new step.
    step_infection_actual <- step_infection_next
```

```
    # Recover of the agent: Before updating again, some agent recover at a rate gamma.
    step_infection_actual <- step_infection_actual*(1 - gamma)


    # We stop the process if everyone is infected
    if(infected_prop[step] %in% c(0,1) ){
      break
    }

  }

  # Return the results
  return(list(infected_count = infected_prop,history = history, steps_needed = step))
}
```

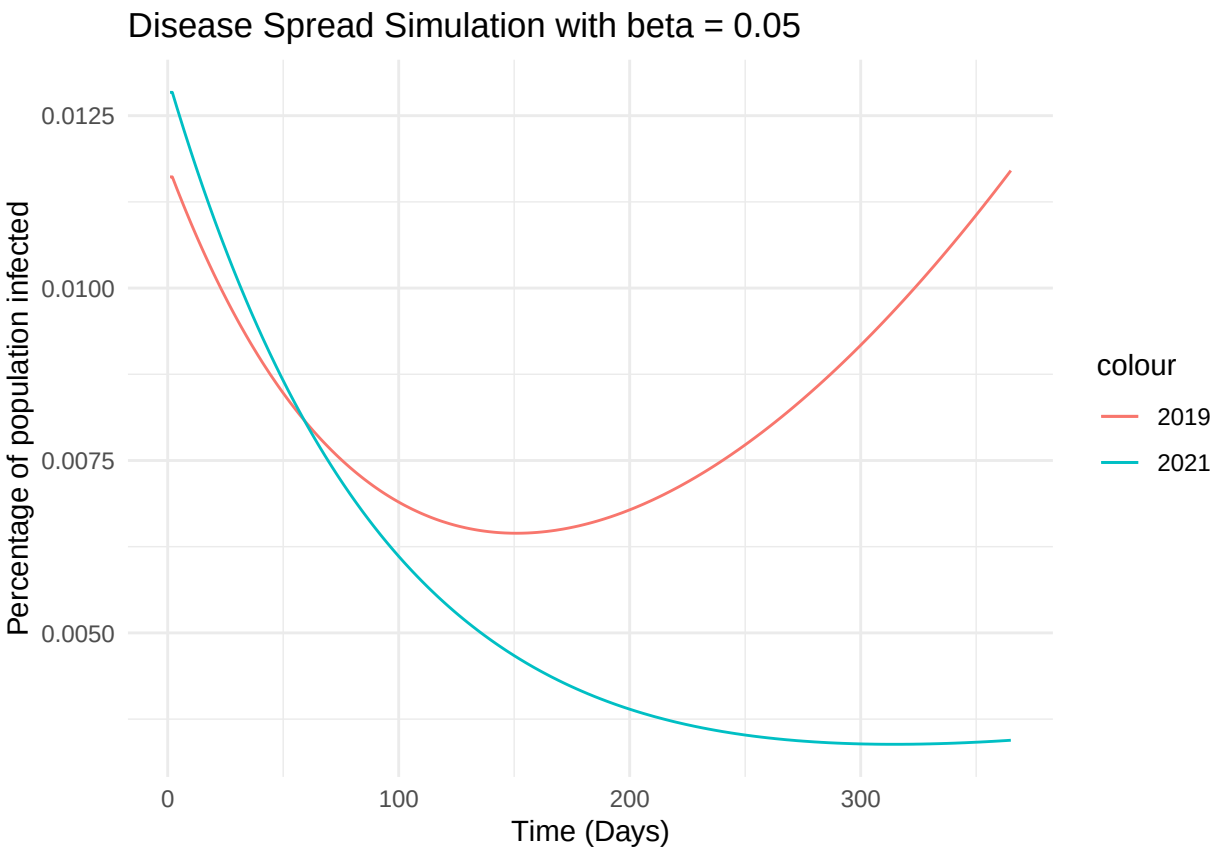
To use this function, we need to provide the graph, the graph of 2019, the transmission rate (beta), the recovery rate (gamma), the number of initial infected agent (n_seeds), and the maximum number of steps for the simulation.

6.1 Low Transmission Rate

We are going to take a transmission rate equal β to 0.05 to model a low contagious disease. We are also taking the recovery the γ equal to 0.01, and the maximum number of steps equal to 365. For the initial infected agents, we are going to choose randomly 15% of the nodes.

```
set.seed(33)

beta <- 0.05
gamma <- 0.01
max_steps <- 365
n_seeds <- 50 #15%

result_low_2019 <- simulate_disease_spread_abm(g_flights_2019,g_flights_2019, beta, gamma, n_seeds, max

result_low_2021 <- simulate_disease_spread_abm(g_flights_2021,g_flights_2019, beta, gamma, n_seeds, max
```

```
ggplot() +
  geom_line(data = data.frame(time = 1:length(result_low_2019$infected_count),
                              n_informed = result_low_2019$infected_count),
            aes(x = time, y = n_informed, color = "2019")) +
  geom_line(data = data.frame(time = 1:length(result_low_2021$infected_count),
                              n_informed = result_low_2021$infected_count),
            aes(x = time, y = n_informed, color = "2021")) +
  ggtitle("Disease Spread Simulation with beta = 0.05") +
  xlab("Time (Days)") +
  ylab("Percentage of population infected") +
  theme_minimal()
```



We see that, as expected, the restrictions of 2021 on air transportation have their effect. The spreading is quickly contained compared to the air transportation without the restrictions.

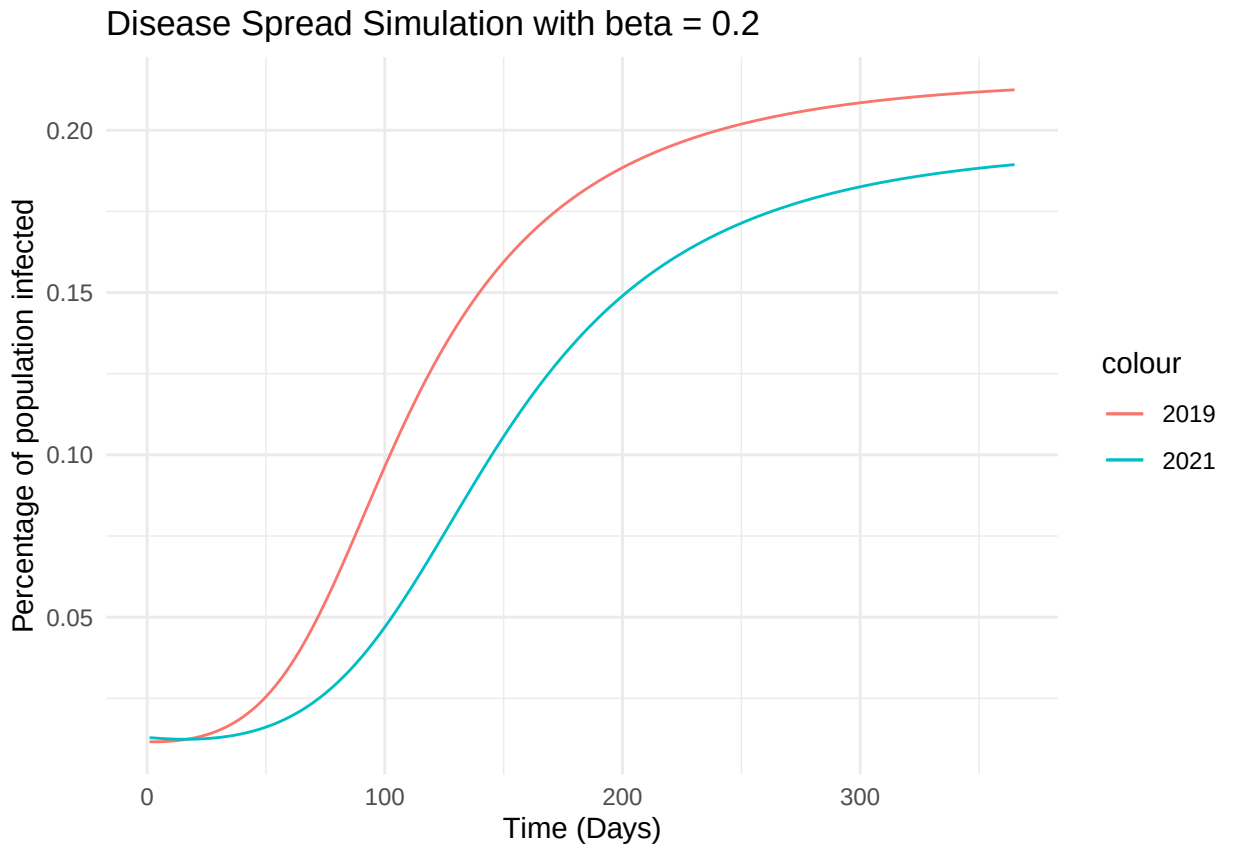
6.2 Medium Transmission Rate

Now, we are going to take a transmission rate equal β to 0.2 to model a medium contagious disease. We are also taking the recovery the γ equal to 0.01, and the maximum number of steps equal to 365. For the initial infected agents, we are going to choose randomly 15% of the nodes.

```
set.seed(33)

beta <- 0.2
gamma <- 0.01
max_steps <- 365
n_seeds <- 50

result_med_2019 <- simulate_disease_spread_abm(g_flights_2019,g_flights_2019, beta, gamma, n_seeds, max

result_med_2021 <- simulate_disease_spread_abm(g_flights_2021,g_flights_2019, beta, gamma, n_seeds, max
```

```
ggplot() +
  geom_line(data = data.frame(time = 1:length(result_med_2019$infected_count),
                              n_informed = result_med_2019$infected_count),
            aes(x = time, y = n_informed, color = "2019")) +
  geom_line(data = data.frame(time = 1:length(result_med_2021$infected_count),
```

```
                              n_informed = result_med_2021$infected_count),
          aes(x = time, y = n_informed, color = "2021")) +
  ggtitle("Disease Spread Simulation with beta = 0.2") +
  xlab("Time (Days)") +
  ylab("Percentage of population infected") +
  theme_minimal()
```

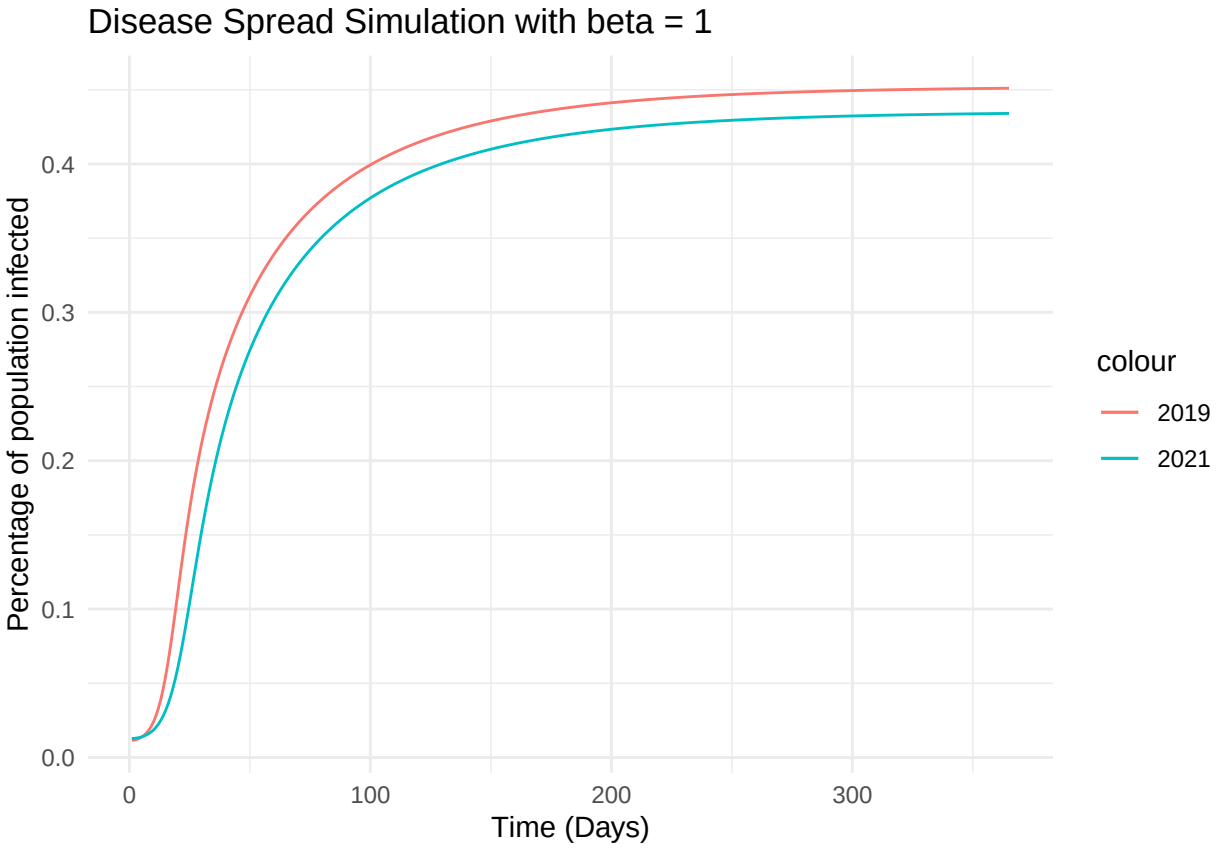


6.3 High Transmission Rate

And finally, we are going to take a transmission rate equal β to 1 to model a highly contagious disease. We are also taking the recovery the γ equal to 0.01, and the maximum number of steps equal to 365. For the initial infected agents, we are going to choose randomly 15% of the nodes.

```
set.seed(33)

beta <- 1
gamma <- 0.01
max_steps <- 365
n_seeds <- 50

result_high_2019 <- simulate_disease_spread_abm(g_flights_2019,g_flights_2019, beta, gamma, n_seeds, ma

result_high_2021 <- simulate_disease_spread_abm(g_flights_2021,g_flights_2019, beta, gamma, n_seeds, ma
```

```r
ggplot() +
  geom_line(data = data.frame(time = 1:length(result_high_2019$infected_count),
                              n_informed = result_high_2019$infected_count),
            aes(x = time, y = n_informed, color = "2019")) +
  geom_line(data = data.frame(time = 1:length(result_high_2021$infected_count),
                              n_informed = result_high_2021$infected_count),
            aes(x = time, y = n_informed, color = "2021")) +
  ggtitle("Disease Spread Simulation with beta = 1") +
  xlab("Time (Days)") +
  ylab("Percentage of population infected") +
  theme_minimal()
```



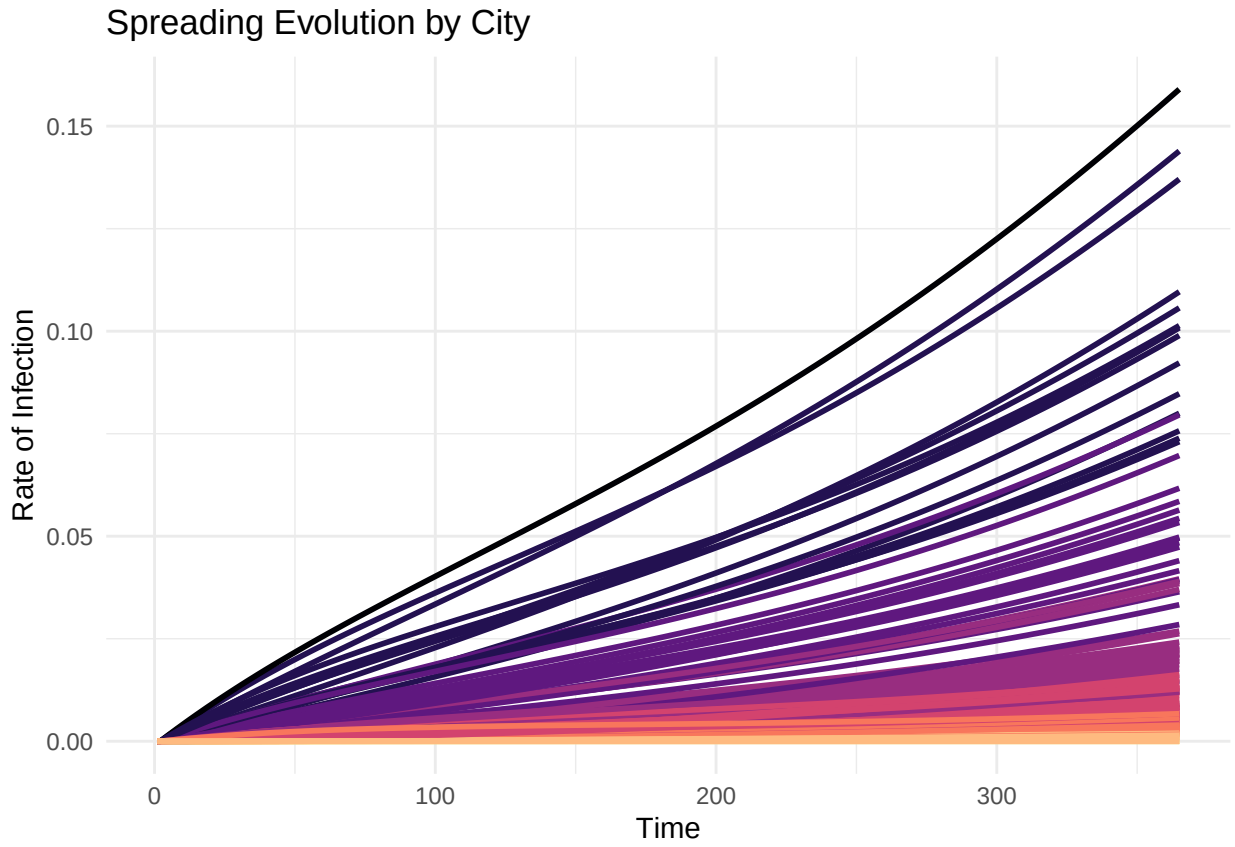
7 - Predictive information of the strength centrality

In this part, we plot the rate of infection by city. The gradient corresponds to the level of the strength centrality. The more central the darker, the less central the lighter. We are doing this plot for each year and situation.

7.1 Low Transmission Rate

Situation in 2019

```
# Centrality normalized
centrality_2019 <- strength(g_flights_2019, V(g_flights_2019), mode = "in", weights = E(g_flights_2019)$

City_evolution <- do.call(rbind.data.frame,result_low_2019$history)
colnames(City_evolution) <- colnames(as.matrix(igraph::get.adjacency(g_flights_2019,attr = "weight")))
City_evolution$time <- c(1:max_steps)

gg <- ggplot(City_evolution, aes(x = time)) +
  theme_minimal() +  # Or any other theme of your choice
  labs(title = "Spreading Evolution by City", x = "Time", y = "Rate of Infection")
ToADD <-  colnames(City_evolution)[which(as.numeric(City_evolution[1,]) ==0)]
ncolor <- max(ceiling(log(centrality_2019[which(as.numeric(City_evolution[1,]) == 0)] + 1)))

# Add lines for each city
for (i in ToADD) {
  gg <- gg +
    geom_line(aes(y = !!sym(i)), color =rev(magma(ncolor))[ceiling(log(centrality_2019[i]+1))],linewidth
}
gg
```



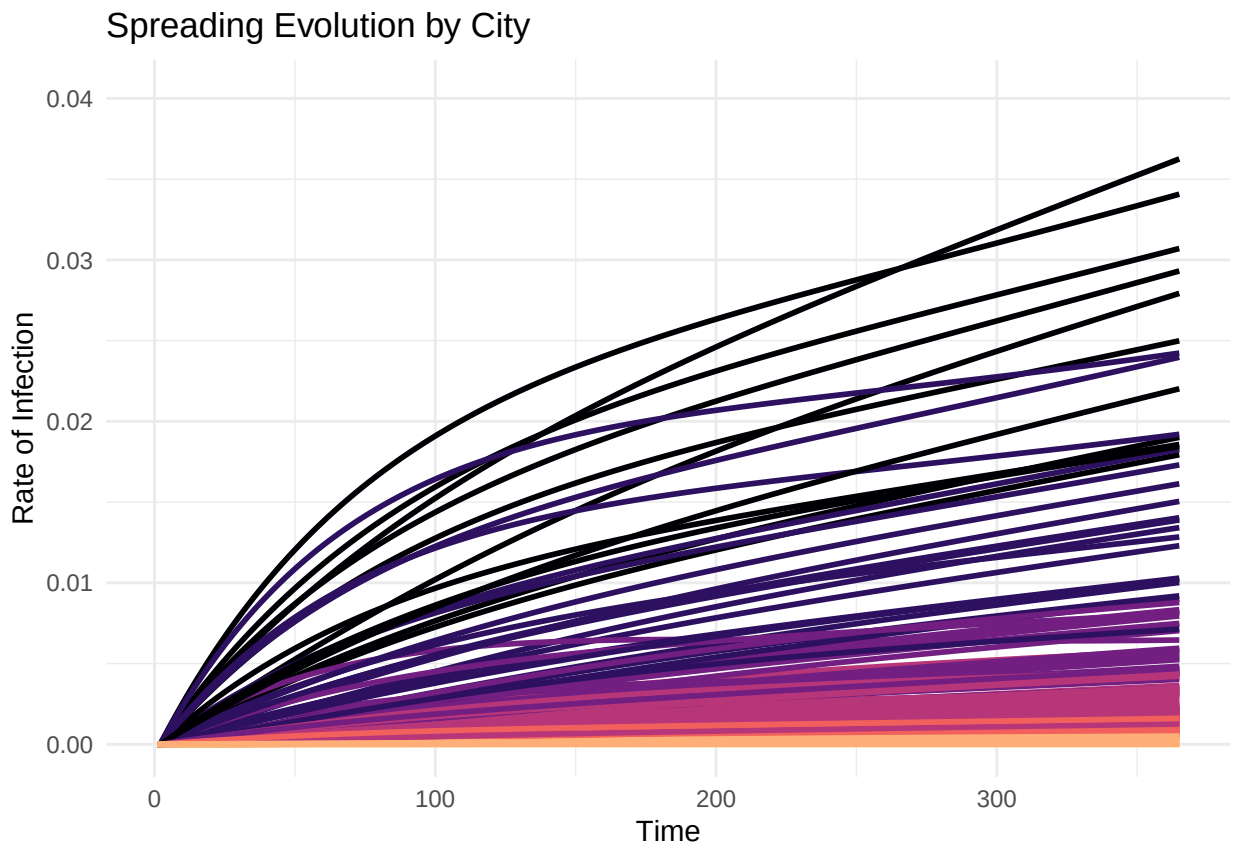
It is quite clear that the centrality has a predictive power in the disease spreading. We see that the cities with the most central airport are the most affected, even for a low contagious disease. On the other hand,

the cities with non-central airports in yellow are not affected at all because they are a bit disconnected. Thus it confirms that the spreading first comes to the most central airport. However, the scale is very low, we talk about 4% of affected people at a maximum.

Situation in 2021

```
# Centrality normalized
centrality_2021 <- strength(g_flights_2021, V(g_flights_2021), mode = "in", weights = E(g_flights_2021)$

City_evolution <- do.call(rbind.data.frame,result_low_2021$history)
colnames(City_evolution) <- colnames(as.matrix(igraph::get.adjacency(g_flights_2021,attr = "weight")))
City_evolution$time <- c(1:max_steps)

# Only add the cities that were not infected initially :
gg <- ggplot(City_evolution, aes(x = time)) +
  theme_minimal() +  # Or any other theme of your choice
  labs(title = "Spreading Evolution by City", x = "Time", y = "Rate of Infection")
ToADD <-  colnames(City_evolution)[which(as.numeric(City_evolution[1,]) ==0)]
ncolor <- max(ceiling(log(centrality_2021[which(as.numeric(City_evolution[1,]) ==0)]+1)))

# Add lines for each city
for (i in ToADD) {
  gg <- gg +
    geom_line(aes(y = !!sym(i)), color =rev(magma(ncolor))[ceiling(log(centrality_2019[i]+1))],linewidth
}
gg
```

```
## Warning: Removed 365 rows containing missing values (`geom_line()`).
```

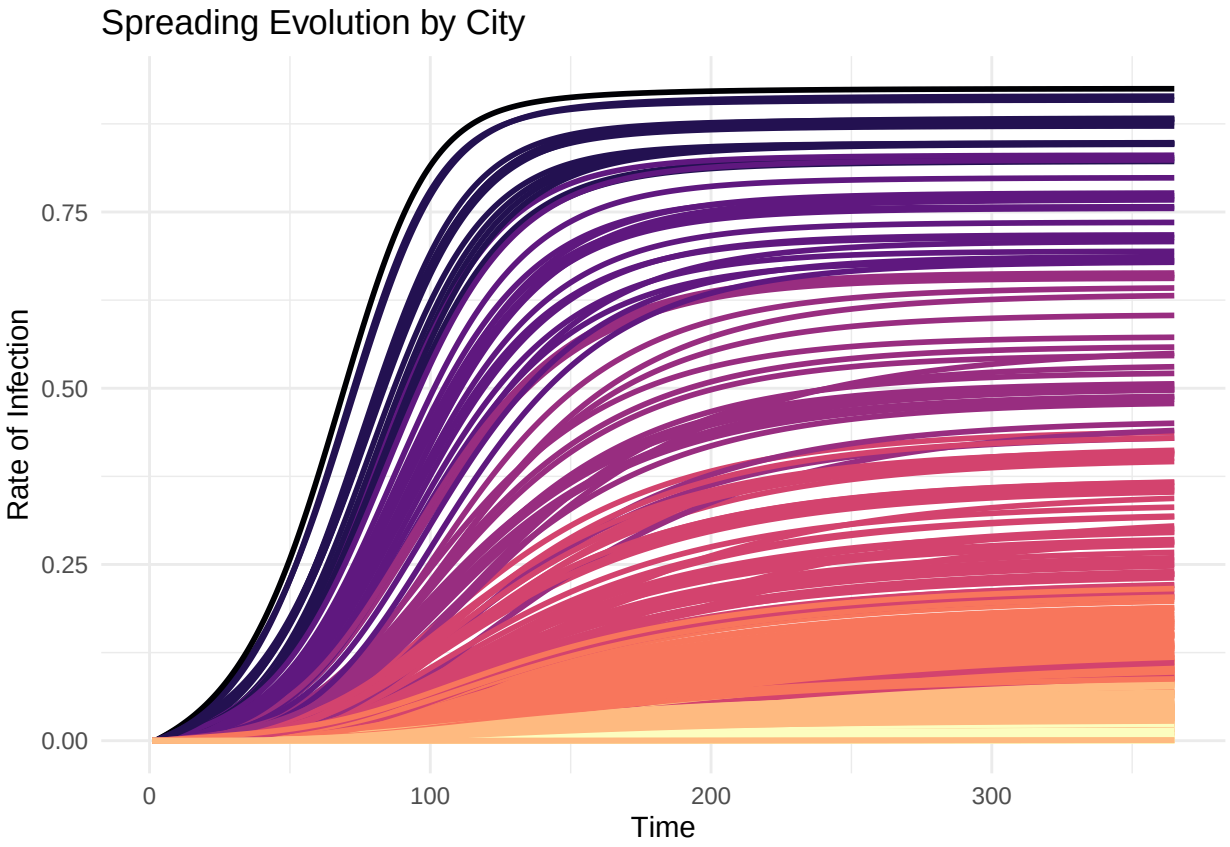


Here, the restriction has a very good effect, the level of infection is lower than without restriction.

7.2 Medium Transmission Rate

Situation in 2019

```
# Centrality normalized
centrality_2019 <- strength(g_flights_2019, V(g_flights_2019), mode = "in", weights = E(g_flights_2019)$

City_evolution <- do.call(rbind.data.frame,result_med_2019$history)
colnames(City_evolution) <- colnames(as.matrix(igraph::get.adjacency(g_flights_2019,attr = "weight")))
City_evolution$time <- c(1:max_steps)

gg <- ggplot(City_evolution, aes(x = time)) +
  theme_minimal() +  # Or any other theme of your choice
  labs(title = "Spreading Evolution by City", x = "Time", y = "Rate of Infection")
ToADD <-  colnames(City_evolution)[which(as.numeric(City_evolution[1,]) ==0)]
ncolor <- max(ceiling(log(centrality_2019[which(as.numeric(City_evolution[1,]) ==0)]+1)))

# Add lines for each city
for (i in ToADD) {
  gg <- gg +
    geom_line(aes(y = !!sym(i)), color =rev(magma(ncolor))[ceiling(log(centrality_2019[i]+1))],linewidth
}
gg
```



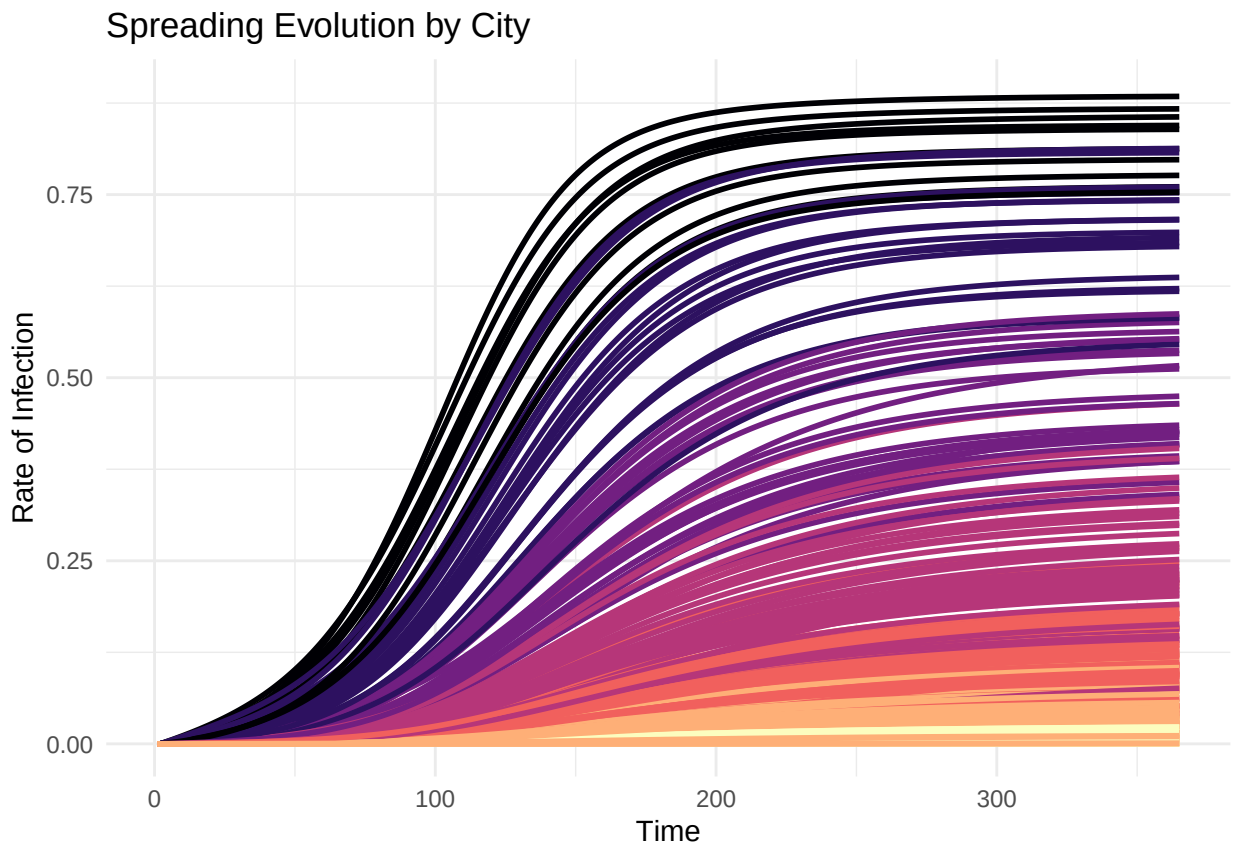
The cities with the major airports are the ones responsible for the spreading, it is quite clear here, that they are really affected.

Situation in 2021

```
# Centrality normalized
centrality_2021 <- strength(g_flights_2021, V(g_flights_2021), mode = "in", weights = E(g_flights_2021)$
City_evolution <- do.call(rbind.data.frame,result_med_2021$history)
colnames(City_evolution) <- colnames(as.matrix(igraph::get.adjacency(g_flights_2021,attr = "weight")))
City_evolution$time <- c(1:max_steps)


gg <- ggplot(City_evolution, aes(x = time)) +
  theme_minimal() +  # Or any other theme of your choice
  labs(title = "Spreading Evolution by City", x = "Time", y = "Rate of Infection")
ToADD <-  colnames(City_evolution)[which(as.numeric(City_evolution[1,]) ==0)]
ncolor <- max(ceiling(log(centrality_2021[which(as.numeric(City_evolution[1,]) ==0)]+1)))

# Add lines for each city
for (i in ToADD) {
  gg <- gg +
    geom_line(aes(y = !!sym(i)), color =rev(magma(ncolor))[ceiling(log(centrality_2019[i]+1))],linewidth
```

```
}
gg
```

```
## Warning: Removed 365 rows containing missing values (`geom_line()`).
```



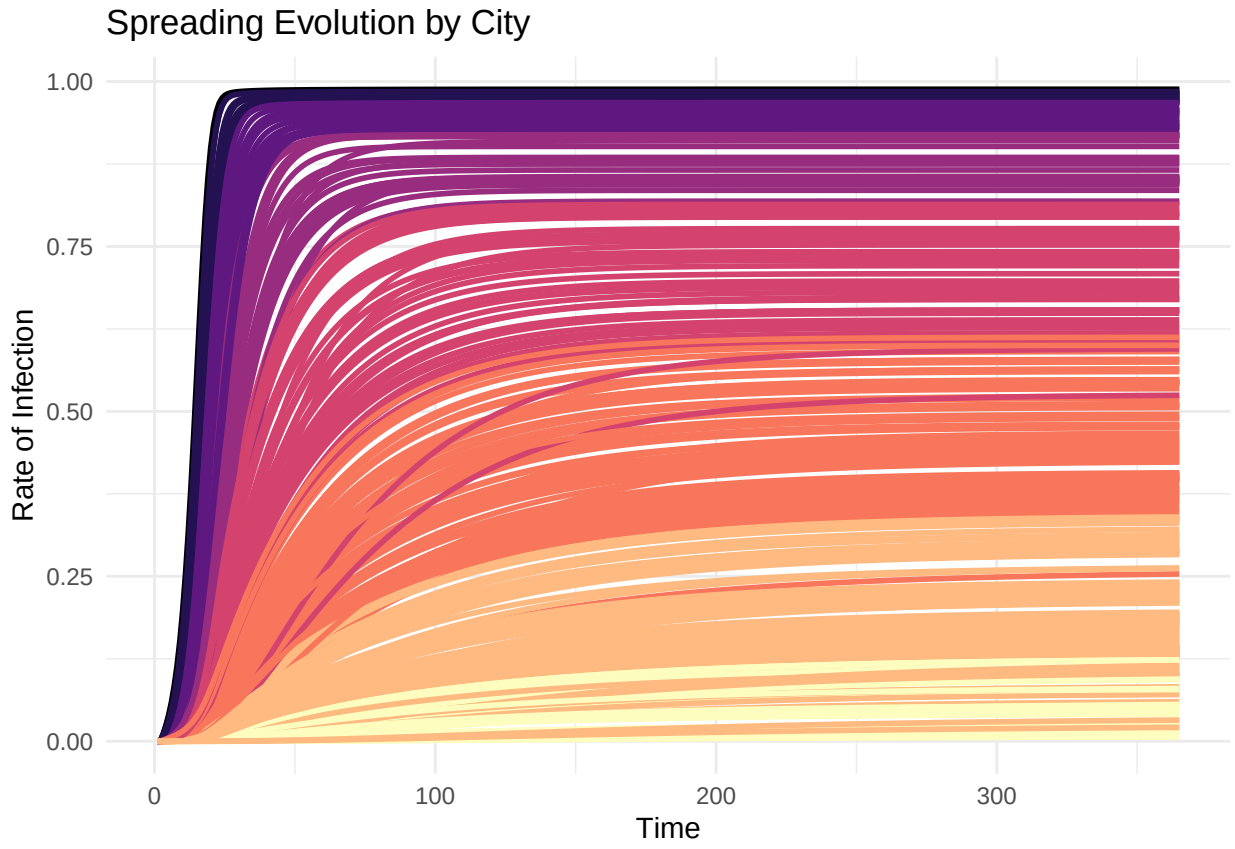
The cities with the major airports are the ones responsible for the spreading, it is quite clear here, that they are really affected but at a slower rate than without restriction.

7.3 High Transmission Rate

Situation in 2019

```
# Centrality normalized
centrality_2019 <- strength(g_flights_2019, V(g_flights_2019), mode = "in", weights = E(g_flights_2019)$
City_evolution <- do.call(rbind.data.frame,result_high_2019$history)
colnames(City_evolution) <- colnames(as.matrix(igraph::get.adjacency(g_flights_2019,attr = "weight")))
City_evolution$time <- c(1:max_steps)


gg <- ggplot(City_evolution, aes(x = time)) +
  theme_minimal() +  # Or any other theme of your choice
  labs(title = "Spreading Evolution by City", x = "Time", y = "Rate of Infection")
ToADD <-  colnames(City_evolution)[which(as.numeric(City_evolution[1,]) ==0)]
```

```
ncolor <- max(ceiling(log(centrality_2019[which(as.numeric(City_evolution[1,]) ==0)]+1)))

# Add lines for each city
for (i in ToADD) {
  gg <- gg +
    geom_line(aes(y = !!sym(i)), color =rev(magma(ncolor))[ceiling(log(centrality_2019[i]+1))],linewidt
}
gg
```

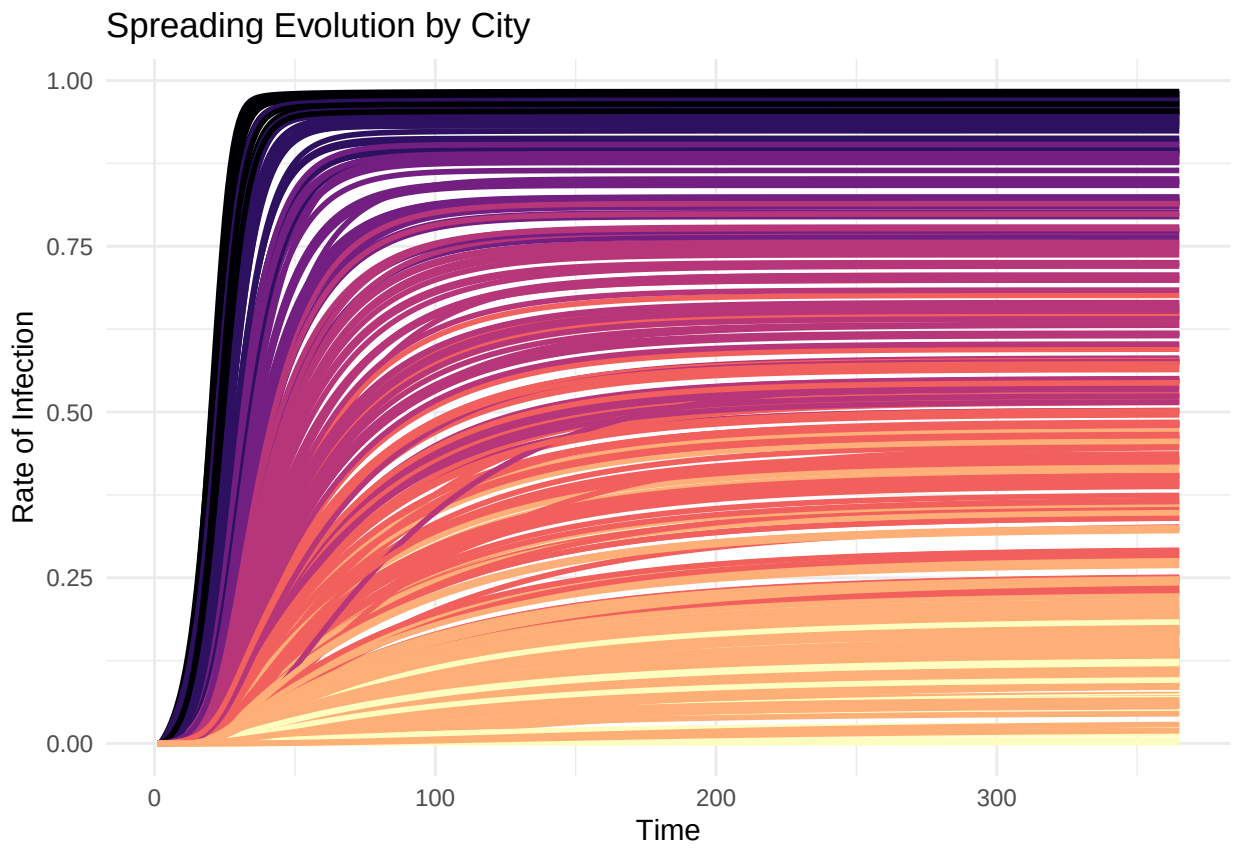


Situation in 2021

```
# Centrality normalized
centrality_2021 <- strength(g_flights_2021, V(g_flights_2021), mode = "in", weights = E(g_flights_2021)$
City_evolution <- do.call(rbind.data.frame,result_high_2021$history)
colnames(City_evolution) <- colnames(as.matrix(igraph::get.adjacency(g_flights_2021,attr = "weight")))
City_evolution$time <- c(1:max_steps)


gg <- ggplot(City_evolution, aes(x = time)) +
  theme_minimal() +  # Or any other theme of your choice
  labs(title = "Spreading Evolution by City", x = "Time", y = "Rate of Infection")
ToADD <-  colnames(City_evolution)[which(as.numeric(City_evolution[1,]) ==0)]
ncolor <- max(ceiling(log(centrality_2021[which(as.numeric(City_evolution[1,]) ==0)]+1)))
```

```
# Add lines for each city
for (i in ToADD) {
  gg <- gg +
    geom_line(aes(y = !!sym(i)), color =rev(magma(ncolor))[ceiling(log(centrality_2019[i]+1))],linewidth
}
gg
```

```
## Warning: Removed 365 rows containing missing values (`geom_line()`).
```



The cities with the major airports are the ones responsible for the spreading, they are quickly affected. The difference in disease spreading with or without restriction is not that clear here. It is like the disease is so terrible that even the 2021 restrictions were not enough. There is only a gain of a few days with the restrictions.