Computational Notebook

Ego Networks & Global Flight Connectivity

Farhan Sadeek

2026-02-21

Contents

Summary

I had to split this computational notebook into two parts, the first part is about my own ego network and the second is about the dataset I picked about airport and the interconnected networks between them. For the first part I analyzed my personal **ego network** using McCabe's framework with the three attributes density, transitivity, betweenness, and modularity to understand how I am connected with different social groups. Since I travel a lot and mostly by air the second is a large-scale **flight network** from global aviation data using a random sample of 500 airports, then I used descriptive analysis techniques from Kolaczyk and Csárdi's *Statistical Analysis of Network Data with R* to understand some partterns in graph and networks.

Ego Network

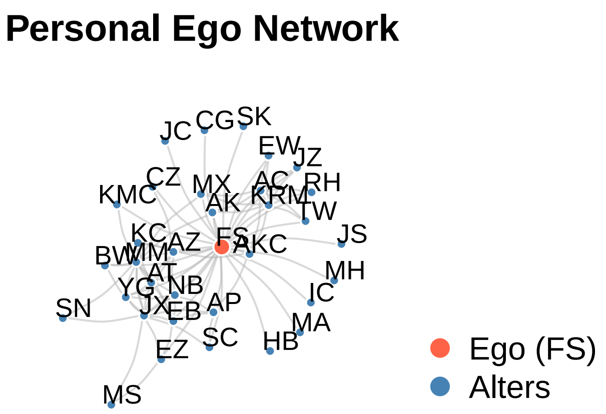
I will start off with the definition of **ego network**. An ego network tries to gather more information about the local neighborhood around a single node. In the ego network of my life, I am the **ego**, its direct connections (the **alters**), and the connections between them. According to McCabe (2016), ego networks are a fundamental unit of social network analysis because they represent the immediate social environment of an individual.

```r
## Reading and building the ego network
ego_net_link =
 ↪  "https://notes.farhansadeek.com/dartmouth/math7/homework/Ego_Network.csv"
ego <- read.csv(ego_net_link)
ego_network <- simplify(graph_from_data_frame(ego, directed =
 ↪  FALSE))
```

Visualizing the Ego Network

```r
## Ego node setup
ego_node <- "FS"

## Color and size: ego vs alters (Claude Code)
```

```r
node_colors <- ifelse(V(ego_network)$name == ego_node, "tomato",
 ↪  "steelblue")
node_sizes <- ifelse(V(ego_network)$name == ego_node, 12, 7)

layout_fr <- layout_with_fr(ego_network)

## Plot the ego network (Claude Code)
plot(ego_network,
     layout = layout_fr,
     vertex.size = node_sizes,
     vertex.color = node_colors,
     vertex.frame.color = "white",
     vertex.label.family = "sans",
     vertex.label.color = "black",
     vertex.label.dist = 1.5,
     vertex.label.cex = 0.8,
     edge.arrow.size = 0.4,
     edge.curved = 0.2,
     edge.color = adjustcolor("gray70", alpha.f = 0.5),
     main = "Personal Ego Network")

legend("bottomright", legend = c("Ego (FS)", "Alters"),
       pt.bg = c("tomato", "steelblue"), col = "white",
       pch = 21, pt.cex = 1.5, bty = "n")
```



In my ego network, I am the node connecting many otherwise disconnected people. If we look at the visualization, then it's clear that I am connected to many alters and alters are not very well connected to themselves. Now, this is very common in ego networks, where the ego serves as a central hub bridging otherwise disconnected groups.

Full Ego Network Measures

Now I will calculate the main structural metrics for the **complete ego network**, which includes all ties between the me and the edges that I am connected to, as well as any connections among the my friends themselves. This would allow us to understand communities and the imapact of me in the ego network formed because of me.

```r
## Computing full ego network measures
full_density <- igraph::edge_density(ego_network)
full_transitivity_global <- igraph::transitivity(ego_network,
 ↪  type = "global")
full_transitivity_ego <- igraph::transitivity(ego_network, type =
 ↪  "local",
                          vids = which(V(ego_network)$name ==
                           ↪  ego_node))
full_betweenness <- igraph::betweenness(ego_network)
full_fc <- igraph::cluster_fast_greedy(ego_network)
full_modularity <- igraph::modularity(full_fc)

## Summary table
full_measures <- data.frame(
  Measure = c("Nodes", "Edges", "Ego Degree (number of alters)",
              "Density", "Global Transitivity",
              "Local Transitivity of Ego",
              "Betweenness Centrality of Ego",
              "Normalized Ego Betweenness",
              "Number of Communities", "Modularity",
              "Ego's Community"),
  Value = c(vcount(ego_network),
            ecount(ego_network),
            igraph::degree(ego_network, v = ego_node),
            round(full_density, 4),
            round(full_transitivity_global, 4),
            round(full_transitivity_ego, 4),
            round(full_betweenness[ego_node], 2),
            round(full_betweenness[ego_node] /
             ↪  max(full_betweenness), 4),
            length(full_fc),
            round(full_modularity, 4),
            membership(full_fc)[ego_node])
)

kable(full_measures, col.names = c("Measure", "Value"), align =
 ↪  c("l", "r"))
```

Measure	Value
Nodes	34.0000
Edges	85.0000
Ego Degree (number of alters)	30.0000
Density	0.1515
Global Transitivity	0.2786
Local Transitivity of Ego	0.0920
Betweenness Centrality of Ego	380.5300
Normalized Ego Betweenness	1.0000
Number of Communities	2.0000
Modularity	0.2989
Ego's Community	2.0000

Since I am the ego I am the center of the network directly connected to almost all other nodes; the network as a whole is moderately dense given its size, but alters have relatively low connectivity amongst themselves, indicated by the comparatively low local transitivity for the ego. My betweenness centrality is maximized showing that I am the main bridge in the network, and most communication flows through me. Since the modularity is high it means that that there might have some clustering among the alters desite me being the center of the network.

Alter-Only Network without Ego

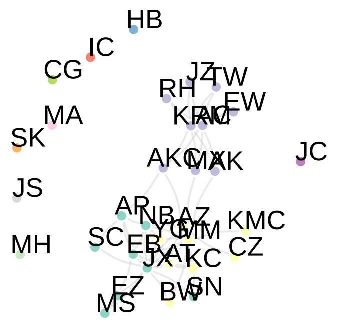
Now I will have to remove the ego node to create the **alter-only induced subgraph** that has only the alter-alter edges. Now, this is important because it shows us how connected the alters are to each other *without* the ego serving as a bridge.

```
## Remove ego to get alter-only network
alter_network <- igraph::delete_vertices(ego_network,
↪  which(V(ego_network)$name == ego_node))
## Alter-only network measures
alter_density <- igraph::edge_density(alter_network)
alter_transitivity_global <- igraph::transitivity(alter_network,
↪  type = "global")
alter_connected <- igraph::is_connected(alter_network)
alter_n_components <- igraph::components(alter_network)$no
alter_fc <- igraph::cluster_louvain(alter_network)
alter_modularity <- igraph::modularity(alter_fc)

alter_measures <- data.frame(
  Measure = c("Nodes", "Edges", "Density", "Global Transitivity",
              "Is Connected", "Number of Components",
              "Number of Communities", "Modularity"),
  Value = c(vcount(alter_network),
            ecount(alter_network),
            round(alter_density, 4),
            round(alter_transitivity_global, 4),
            alter_connected,
            alter_n_components,
            length(alter_fc),
            round(alter_modularity, 4))
)

kable(alter_measures, col.names = c("Measure", "Value"), align =
 ↪  c("l", "r"))
```

Measure	Value
Nodes	33.0000
Edges	55.0000
Density	0.1042
Global Transitivity	0.3666
Is Connected	0.0000
Number of Components	9.0000
Number of Communities	11.0000
Modularity	0.3615

```
## Visualize alter-only network by community
n_communities <- max(alter_fc$membership)
pal <- if (n_communities ⩽ 12) brewer.pal(max(3, n_communities),
 ↪  "Set3") else rainbow(n_communities)
alter_node_colors <- pal[alter_fc$membership]

plot(alter_network,
     vertex.size = 8,
     vertex.color = alter_node_colors,
     vertex.frame.color = "white",
     vertex.label.family = "sans",
     vertex.label.color = "black",
     vertex.label.dist = 1.5,
     vertex.label.cex = 0.8,
     edge.arrow.size = 0.4,
     edge.curved = 0.2,
     edge.color = adjustcolor("gray80", alpha.f = 0.4),
     layout = layout_with_fr(alter_network),
     main = "Alter-Only Network (Colored by Community)")
```

Alter−Only Network (Colored by Community)



Comparison Table

```
results <- data.frame(
  Measure = c("Nodes", "Edges", "Density", "Global Transitivity",
   ↪  "Modularity", "Communities"),
  Full_w_ego = c(vcount(ego_network), ecount(ego_network),
   ↪  full_density, full_transitivity_global, full_modularity,
   ↪  length(full_fc)),
  Alter_only = c(vcount(alter_network), ecount(alter_network),
   ↪  alter_density, alter_transitivity_global, alter_modularity,
   ↪  length(alter_fc))
)

kable(results, col.names = c("Measure", "Full (w/ ego)",
 ↪  "Alter-only"), digits = 4)
```

Measure	Full (w/ ego)	Alter-only
Nodes	34.0000	33.0000
Edges	85.0000	55.0000
Density	0.1515	0.1042
Global Transitivity	0.2786	0.3666
Modularity	0.2989	0.3615
Communities	2.0000	11.0000

McCabe's Network Typology

According to the McCabe, there are three types of network structure - **Tight-knitters** have one densely connected, often exclusive group (high density, high transitivity, low modularity) - **Compartmentalizers** maintain distinct, separate groups that do not mingle (moderate density, high modularity, multiple clear communities). - **Samplers** maintain separate individual or small-group friendships across different areas of life (low density, low transitivity, many components or isolates in the alter-only network).

I can classify my ego network by examining the structural signatures in the alter-only network, since that reveals the true pattern of connections among my contacts without me as the bridge.

```r
## Gemini 3.1 Pro
typology_metrics <- data.frame(
  Metric = c("Alter-only density", "Alter-only transitivity",
             "Alter-only modularity", "Number of communities",
             "Number of components"),
  Value = c(round(alter_density, 4),
            round(alter_transitivity_global, 4),
            round(alter_modularity, 4),
            length(alter_fc),
            alter_n_components)
)

kable(typology_metrics, col.names = c("Metric", "Value"), align =
↪  c("l", "r"),
      caption = "Alter-Only Network Metrics for Typology
      ↪  Classification")
```

Table 4: Alter-Only Network Metrics for Typology Classification

Metric	Value
Alter-only density	0.1042
Alter-only transitivity	0.3666
Alter-only modularity	0.3615
Number of communities	11.0000
Number of components	9.0000

```r
## Classification logic based on McCabe (2016)
if (alter_density > 0.3 && alter_modularity < 0.3) {
  ego_type <- "Tight-knitter"
} else if (alter_density < 0.10 && alter_n_components > 3) {
  ego_type <- "Sampler"
} else {
  ego_type <- "Compartmentalizer"
}
```

Based on these metrics, I classify as a **Compartmentalizer**. Here is the the pattern that I noticed there was

- A **Tight-knitter** would show alter-only density above 0.3 and modularity below 0.3 — one big, tightly connected group where everyone knows everyone.

- A **Sampler** would show very low alter-only density (below 0.15) and many disconnected components (more than 3) — scattered friendships that don't form groups.
- A **Compartmentalizer** falls in between: the alter-only network has moderate density with clear community structure (high modularity) — distinct friend groups (e.g., academic, extracurricular, home) that don't overlap much.

With an alter-only density of 0.1042, modularity of 0.3615, and 9 components, my network fits the **Compartmentalizer** pattern. My contacts cluster into separate social circles — groups from different parts of my life that are internally connected but rarely mingle with each other. When I am removed from the network, these groups become clearly visible as distinct communities.

Alter Role Classification

Now Gemini also classified each alter by their structural role within the network. An alter's degree, local clustering coefficient, and betweenness centrality together reveal whether they sit inside a tight group, serve as a bridge between groups, or are relatively isolated.

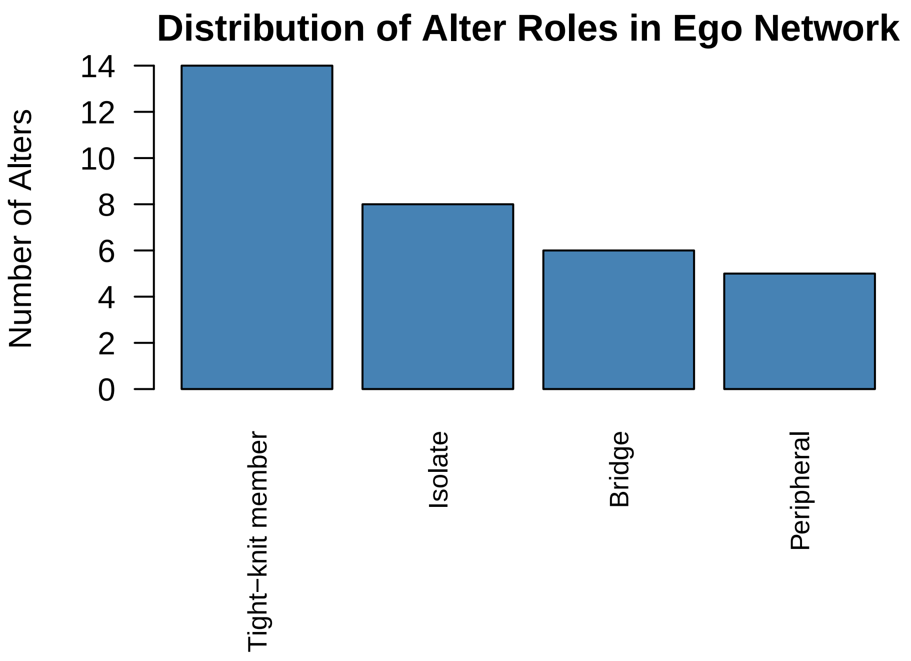
```r
## Classifying each alter by their structural role
alter_names <- V(alter_network)$name
alter_deg <- igraph::degree(alter_network)
alter_local_trans <- igraph::transitivity(alter_network, type =
↪  "local")
alter_betw <- igraph::betweenness(alter_network, normalized =
↪  TRUE)
alter_community <- membership(alter_fc)

alter_classification <- data.frame(
  Alter = alter_names,
  Degree = alter_deg,
  Local_Clustering = round(alter_local_trans, 4),
  Betweenness = round(alter_betw, 4),
  Community = alter_community
)

## Assigning roles based on degree, clustering, and betweenness
alter_classification$Role <- ifelse(
  alter_deg == 0, "Isolate",
  ifelse(alter_betw > median(alter_betw[alter_betw > 0], na.rm =
  ↪  TRUE) &
         alter_deg >= median(alter_deg[alter_deg > 0]),
         "Bridge",
         ifelse(!is.na(alter_local_trans) & alter_local_trans >
         ↪  0.5,
                "Tight-knit member",
                  "Peripheral")))

kable(alter_classification ▷ arrange(desc(Degree)),
      col.names = c("Alter", "Degree", "Local Clustering",
       ↪  "Betweenness",
                     "Community", "Role"),
      align = c("l", "r", "r", "r", "r", "l"),
      caption = "Alter Classification by Network Role")
```

Table 5: Alter Classification by Network Role

	Alter	Degree	Local Clustering	Betweenness	Community	Role
JX	JX	13	0.2692	0.1319	1	Bridge
MM	MM	12	0.2424	0.2092	2	Bridge
KRM	KRM	8	0.2500	0.0862	3	Bridge
AC	AC	8	0.2500	0.0862	3	Bridge
AZ	AZ	7	0.3810	0.0821	2	Bridge
NB	NB	6	0.6000	0.0077	2	Tight-knit member
AT	AT	6	0.6667	0.0042	2	Tight-knit member
EB	EB	6	0.5333	0.0165	2	Tight-knit member
AP	AP	5	0.4000	0.0411	1	Peripheral
KC	KC	4	1.0000	0.0000	2	Tight-knit member
YG	YG	4	0.8333	0.0004	2	Tight-knit member
AKC	AKC	4	0.1667	0.0868	3	Bridge
EZ	EZ	3	0.6667	0.0010	1	Tight-knit member
MX	MX	3	0.3333	0.0549	3	Peripheral
AK	AK	3	0.3333	0.0549	3	Peripheral
MS	MS	2	1.0000	0.0000	1	Tight-knit member
SC	SC	2	1.0000	0.0000	1	Tight-knit member
BW	BW	2	1.0000	0.0000	1	Tight-knit member
TW	TW	2	1.0000	0.0000	3	Tight-knit member
EW	EW	2	1.0000	0.0000	3	Tight-knit member

	Alter	Degree	Local Clustering	Betweenness	Community	Role
RH	RH	2	1.0000	0.0000	3	Tight-knit member
JZ	JZ	2	1.0000	0.0000	3	Tight-knit member
SN	SN	2	1.0000	0.0000	1	Tight-knit member
CZ	CZ	1	NaN	0.0000	2	Peripheral
KMC	KMC	1	NaN	0.0000	2	Peripheral
IC	IC	0	NaN	0.0000	4	Isolate
HB	HB	0	NaN	0.0000	5	Isolate
SK	SK	0	NaN	0.0000	6	Isolate
CG	CG	0	NaN	0.0000	7	Isolate
MA	MA	0	NaN	0.0000	8	Isolate
JS	JS	0	NaN	0.0000	9	Isolate
JC	JC	0	NaN	0.0000	10	Isolate
MH	MH	0	NaN	0.0000	11	Isolate

```r
par(mar = c(7, 5, 2, 3))  # increase bottom margin
role_summary <- table(alter_classification$Role)
barplot(sort(role_summary, decreasing = TRUE),
        col = "steelblue",
        las = 2,
        cex.names = 0.8,
        ylab = "Number of Alters",
        main = "Distribution of Alter Roles in Ego Network")
```



The alter role distribution reinforces the Compartmentalizer classification. **Isolates** are alters who have no connections to anyone else in my network — they know only me, which is characteristic of sampler-type relationships. **Bridges** are alters with high betweenness who connect different groups, much like I do as the ego. **Tight-knit members** are embedded within a dense cluster where their neighbors are also connected to each other. **Peripheral** alters have some connections but don't fit neatly into a tight group or bridging role.

Comparison and Discussion

Now if we compare the network with and without me then there are a few interesting patterns. The density drops noticeably when I was removed, and that makes sense because I am connected to every alter by definition. Transitivity also changes, meaning that many of my alters know each other only through me. The modularity in the alter-only network is higher, indicating that without me bridging the groups, the alters cluster into more distinct communities such as friend groups from different parts of my life (college, work, hometown) that have little overlap. Now, this is *consistent with McCabe's observation that ego removal often reveals the brokerage role the ego plays*. Now the betweenness centrality in the full network makes sure that I am a middle-man when connecting groups that would otherwise be disconnected.

Flight Network Analysis

Sampling from a Large Dataset

Now, this is the second part of the computational notebook where I am taking a **random sample of 500 airports** from the global flight data. This gives us a more realistic and structurally interesting network with regional and smaller airports alongside major hubs. The network should show a variety of degree distribution and hub-and-spoke topology that is characteristic of real-world modern air transportation networks.

I read a single month of global flight data (April 2020) and then drew my sample.

```
## Loading April 2020 flight data
df <- read.csv("dataset/flightlist_20200401_20200430.csv")

## Counting flights per airport
origin_counts <- df ▷ count(origin, name = "flights") ▷
↪  rename(airport = origin)
dest_counts <- df ▷ count(destination, name = "flights") ▷
↪  rename(airport = destination)
airport_activity <- bind_rows(origin_counts, dest_counts) ▷
  group_by(airport) ▷
  summarise(total_flights = sum(flights)) ▷
  arrange(desc(total_flights))

## Remove airports with empty or NA codes
airport_activity <- airport_activity ▷ filter(airport ≠ "" &
↪  !is.na(airport))
```

I used a stratified random sampling approach to ensure the 500-airport sample includes a realistic mix with the very busiest hubs (so the network stays connected) alongside a random draw from the rest. This mirrors how real airline networks work a few major hubs connect to many smaller airports.

```
## Randomly sample 500 airports
set.seed(42)

all_airports <- airport_activity$airport
sampled_airports <- sample(all_airports, min(2000,
↪  length(all_airports)))

## Filter to flights between sampled airports
sampled_df <- df ▷ filter(origin %in% sampled_airports &
↪  destination %in% sampled_airports)

cat("Number of sampled airports:", length(sampled_airports), "\n")
```

```
Number of sampled airports: 2000
```

```r
cat("Number of flights between sampled airports:",
 ↪  nrow(sampled_df), "\n")
```

```
Number of flights between sampled airports: 33675
```

Building the Network

```r
## Selecting relevant columns
sampled_df <- sampled_df ▷
  select(origin, destination, latitude_1, longitude_1, latitude_2,
   ↪  longitude_2) ▷
  drop_na()

## Edge list: weighted by flight count per route
edges <- sampled_df ▷
  group_by(origin, destination) ▷
  summarise(weight = n(), .groups = "drop")

## Vertex list: unique airports with coordinates
origins <- sampled_df ▷
  select(name = origin, lat = latitude_1, long = longitude_1)

destinations <- sampled_df ▷
  select(name = destination, lat = latitude_2, long = longitude_2)

nodes <- bind_rows(origins, destinations) ▷
  distinct(name, .keep_all = TRUE) ▷
  na.omit()

## Building the directed graph
flight_network <- graph_from_data_frame(d = edges, vertices =
 ↪  nodes, directed = TRUE)
flight_network <- simplify(flight_network, remove.multiple =
 ↪  TRUE, remove.loops = TRUE)

## Undirected version for symmetric analyses
flight_undirected <- igraph::as.undirected(flight_network, mode =
 ↪  "collapse")

cat("Directed network:\n")
```

```
Directed network:
```

```r
print(summary(flight_network))
```

```
IGRAPH 1a969c1 DNW- 1342 3713 --
+ attr: name (v/c), lat (v/n), long (v/n), weight (e/n)
IGRAPH 1a969c1 DNW- 1342 3713 --
+ attr: name (v/c), lat (v/n), long (v/n), weight (e/n)
+ edges from 1a969c1 (vertex names):
 [1] OTHH->VABB OTHH->KIAD OTHH->RJBB OTHH->LEMD OTHH->KJFK OTHH-
>VTBS
 [7] OTHH->EBLG OTHH->KDFW OTHH->EGSS OTHH->LIRF OTHH->WSAP OTHH-
>GMMN
[13] OTHH->LOWW OTHH->UKBB OTHH->LEBL OTHH->VOMM OTHH->ENKJ OTHH-
>EDDT
[19] OTHH->RJAA OTHH->EINN OTHH->LKVO OTHH->FARA OTHH->UKKT VABB-
>OTHH
[25] VABB->EBLG VABB->WSAP VABB->VOMM VABB->VTBD VABB->VAPO VABB-
>VEPI
[31] KIAD->OTHH KIAD->LEMD KIAD->KJFK KIAD->KCLT KIAD->KCLE KIAD-
>KDFW
[37] KIAD->KDAL KIAD->KMKE KIAD->KFCM KIAD->KTPA KIAD->KRVS KIAD-
>KLAS
[43] KIAD->KDTW KIAD->03PS KIAD->KX04 KIAD->KFTW KIAD->KBUY KIAD-
>KORF
+ ... omitted several edges
```

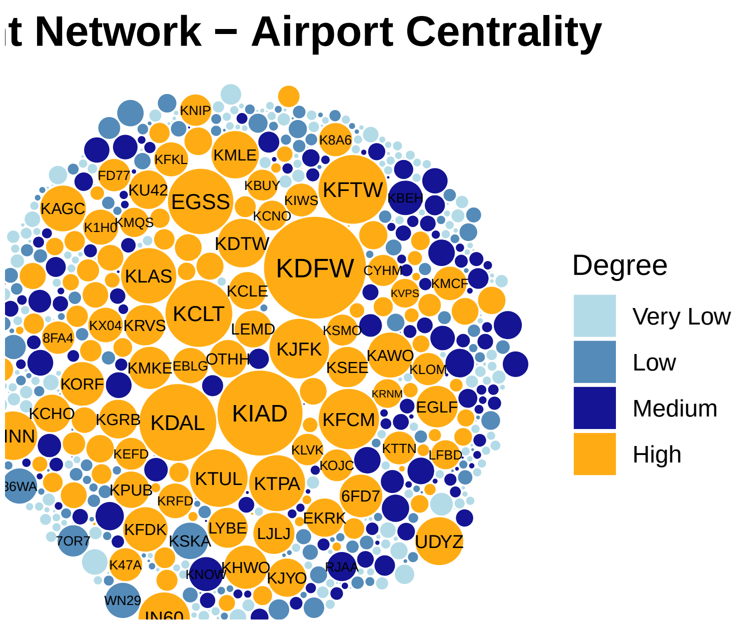
I selected only the columns needed for the analysis, constructed edge and vertex lists, and built both directed and undirected versions of the graph. I then simplified the graph so that there are no multi-edges or self-loops (the edge list was already aggregated by route, but simplification ensures a simple graph and removes any self-loops).

Network Visualization

I visualized the network using a force-directed layout. In a network this large, raw plots can become unreadable, so I used vertex size scaled by degree and edge transparency to highlight the hub-and-spoke structure. I also applied the Fruchterman-Reingold layout algorithm (Fruchterman & Reingold, 1991), which tends to place highly-connected nodes centrally.

```
## Claude Code generated visualization with packcircles for better
 ↪  layoutated
deg <- igraph::degree(flight_undirected)
btw <- igraph::betweenness(flight_undirected)

df <- data.frame(
  airport = V(flight_undirected)$name,
  centrality = btw,
  degree = deg
)

# Aggressive cleaning
df <- df[complete.cases(df), ]
df <- df[df$centrality > 0, ]
df <- df[is.finite(df$centrality), ]
```

```
df$centrality <- as.numeric(df$centrality)

color_pal <- colorRampPalette(c("lightblue", "steelblue",
 ↪  "darkblue", "orange", "red"))(5)
## Use unique breaks so cut() never gets duplicate break points
 ↪  (e.g. when degree has little variation)
deg_breaks <- unique(quantile(df$degree, probs = seq(0, 1,
 ↪  length.out = 6), na.rm = TRUE))
if (length(deg_breaks) ⩾ 2) {
  df$deg_bin <- as.integer(cut(df$degree, breaks = deg_breaks,
 ↪  include.lowest = TRUE, labels = FALSE))
  df$deg_bin <- pmin(df$deg_bin, 5)  # cap at 5 for color_pal
} else {
  df$deg_bin <- 1L
}
df$color   <- color_pal[df$deg_bin]
df$label   <- ifelse(df$centrality ⩾ quantile(df$centrality,
 ↪  0.90), df$airport, "")

# Pack and check for NAs before plotting
packing <- circleProgressiveLayout(df$centrality, sizetype =
 ↪  "area")

# Drop any rows where packing produced NAs
bad <- apply(packing, 1, function(r) any(is.na(r)))
df     <- df[!bad, ]
packing <- packing[!bad, ]

df <- cbind(df, packing)
dat.gg <- circleLayoutVertices(packing, npoints = 100)
dat.gg$deg_bin <- rep(df$deg_bin, each = 101)

ggplot() +
  geom_polygon(data = dat.gg,
               aes(x, y, group = id, fill = factor(deg_bin)),
               colour = "white", linewidth = 0.3, alpha = 0.92) +
  scale_fill_manual(
    values = setNames(color_pal, as.character(1:5)),
    labels = c("Very Low", "Low", "Medium", "High", "Very High"),
    name = "Degree"
  ) +
geom_text(data = df[df$label ≠ "", ],
            aes(x, y, label = label, size = centrality),
            color = "black",show.legend = FALSE) +
  scale_size_continuous(range = c(1.5, 3.5)) +
  coord_equal() +
  theme_void() +
  theme(
    legend.position = "right",
    plot.title = element_text(hjust = 0.5, size = 16, face =
     ↪   "bold"),
    plot.background = element_rect(fill = "white", color = NA)
  ) +
  labs(title = "Flight Network – Airport Centrality")
```



From this visualization I can see the hub-and-spoke structure that airlines follow which is a small number of airports (colored in red/orange) sit at the center of the layout with many connections, while the majority of airports cluster around the periphery with only a few routes each.

Basic Graph Properties

Below are some basic network property checks designed with the help of Claude Code. These provide an overview of the network's structure.

```
wc <- igraph::clusters(flight_network, mode = "weak")

basic_props <- data.frame(
  Property = c("Number of airports (vertices)",
               "Number of flight routes (edges)",
               "Is the graph simple?",
               "Is weakly connected?",
                  "Is strongly connected?",
                  "Number of weakly connected components",
                  "Size of largest component",
                  "Diameter (unweighted)",
                  "Average path length",
                  "Edge density"),
  Value = c(vcount(flight_network),
            ecount(flight_network),
            is_simple(flight_network),
            is_connected(flight_network, mode = "weak"),
            is_connected(flight_network, mode = "strong"),
            wc$no,
            max(wc$csize),
            diameter(flight_network, weights = NA),
            round(mean_distance(flight_network), 4),
            round(edge_density(flight_network), 6))
)

kable(basic_props, col.names = c("Property", "Value"), align =
 ↪  c("l", "r"))
```

Property	Value
Number of airports (vertices)	1.3420e+03
Number of flight routes (edges)	3.7130e+03
Is the graph simple?	1.0000e+00
Is weakly connected?	0.0000e+00
Is strongly connected?	0.0000e+00
Number of weakly connected components	1.1700e+02
Size of largest component	1.1370e+03
Diameter (unweighted)	1.6000e+01
Average path length	7.7486e+00
Edge density	2.0630e-03

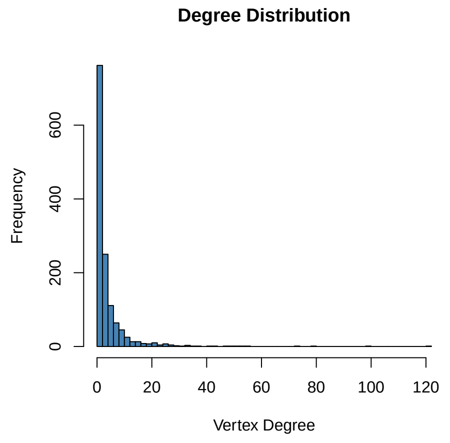
Vertex and Edge Characteristics

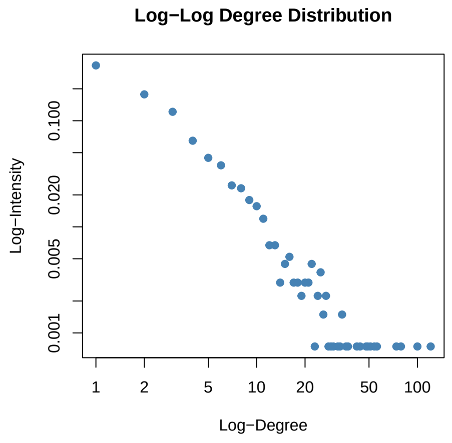
Degree Distribution

Now we will take a look at the degree distribution, which is a very important property of the network. The degree distribution tells us how many connections each airport has. In airline networks, we often see a highly skewed degree distribution where a few major hubs have many connections, while most airports have only a few routes.

```
par(mfrow = c(1, 2))
```

```
## Histogram of degree
hist(igraph::degree(flight_undirected),
     col = "steelblue",
     breaks = 50,
     xlab = "Vertex Degree",
     ylab = "Frequency",
     main = "Degree Distribution")

## Log-log degree distribution to check for power-law behavior
dd.flights <- degree_distribution(flight_undirected)
d <- 0:(length(dd.flights) - 1)
ind <- (dd.flights ≠ 0)
plot(d[ind], dd.flights[ind],
     log = "xy",
     col = "steelblue",
     pch = 19,
     xlab = "Log-Degree",
     ylab = "Log-Intensity",
     main = "Log-Log Degree Distribution")
```





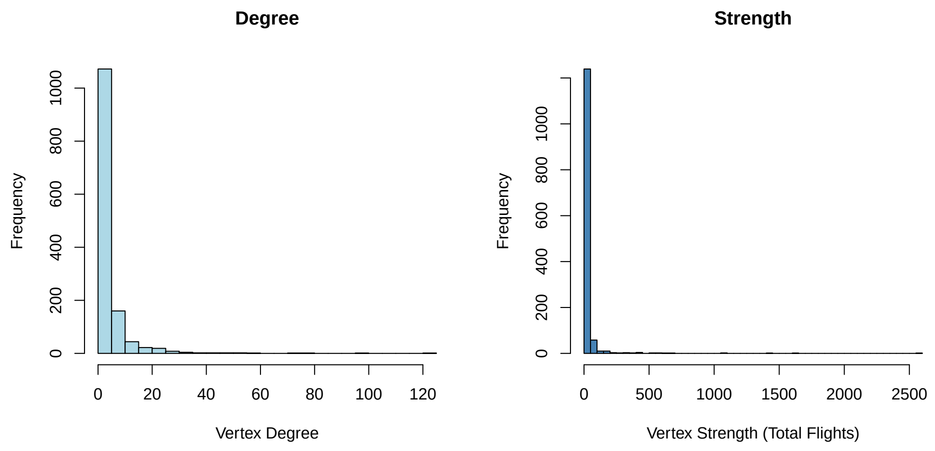
The degree distribution is very skewed, because of the airports don't have very many connections, but the hubs have a lot of connections. The log-long plot shows a linear relationship in the tail, which is an indication of a power-law or scale-free degree distribution. Now we can derive becuase most airlines want to reduce cost and having only a few hubs make it easier for repair and maintenace.

Vertex Strength

While degree counts the number of routes, vertex strength accounts for edge weights which could be represented the number of flights on each route. This could help us find airpots where are people flying a lot more frequently.

```
par(mfrow = c(1, 2))
```

```
hist(igraph::degree(flight_undirected), col = "lightblue",
     xlab = "Vertex Degree", ylab = "Frequency", main = "Degree",
     breaks = 40)

hist(strength(flight_undirected), col = "steelblue",
     xlab = "Vertex Strength (Total Flights)", ylab = "Frequency",
     main = "Strength", breaks = 40)
```

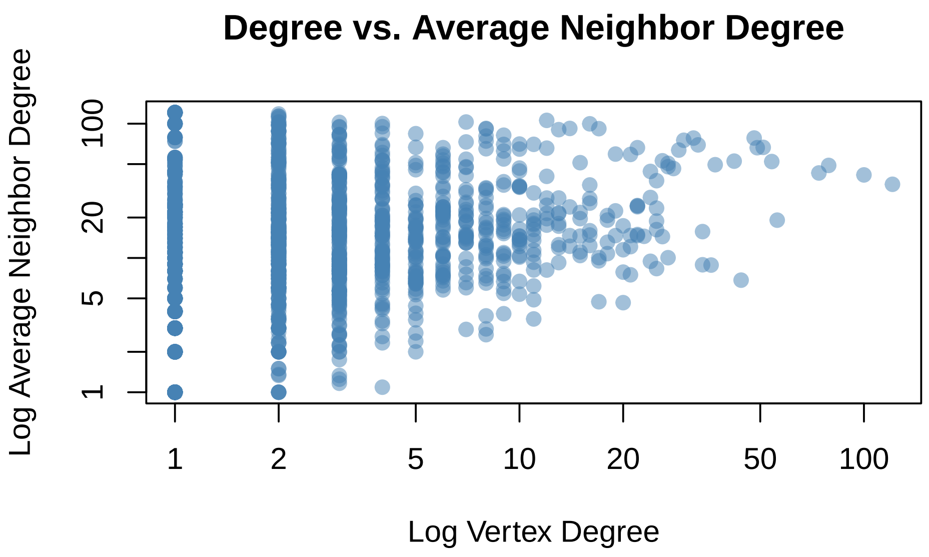


Both distributions are right-skewed, but the strength distribution has an even longer tail. Now this means that not only that the major hubs have more destinations those flights are much more frequent than the routes themselves.

Average Neighbor Degree

Let's take a look at how each airport's number of connections relates to the average number of connections its neighboring airports have. This helps us see if airports with lots of connections mostly link to other well-connected airports, or if they're more likely to connect to smaller, less connected ones.

```
a.nn.deg.flight <- knn(flight_undirected,
 ↪  V(flight_undirected))$knn
plot(igraph::degree(flight_undirected), a.nn.deg.flight,
     log = "xy",
     col = adjustcolor("steelblue", alpha.f = 0.5),
     pch = 19,
     xlab = "Log Vertex Degree",
     ylab = "Log Average Neighbor Degree",
     main = "Degree vs. Average Neighbor Degree")
```



The plot shows a negative trend: higher-degree airports (the major hubs) tend to be connected to neighbors with lower average degree. This is textbook disassortative mixing, which is characteristic of hub-and-spoke transportation networks. The big hubs connect to many small regional airports, which in turn have the hub as their most prominent neighbor. This pattern contrasts with social networks, which are typically assortative (popular people befriend other popular people).

Network Cohesion

Now, this an interesting property to look at how the vertex and edges are connected and the size of the largest component to understand how the network is built.

Connectivity and Components

```
## Vertex and edge connectivity
v_conn <- igraph::vertex_connectivity(flight_undirected)
e_conn <- igraph::edge_connectivity(flight_undirected)

comp <- igraph::clusters(flight_undirected)  # use clusters() to
 ↪  avoid the conflict

cohesion_props <- data.frame(
  Property = c("Vertex connectivity",
               "Edge connectivity",
               "Number of components",
               "Size of largest component",
               "Number of isolates (degree 0)"),
  Value = c(v_conn,
            e_conn,
            comp$no,
            max(comp$csize),
            sum(igraph::degree(flight_undirected) == 0))
)

kable(cohesion_props, col.names = c("Property", "Value"), align =
 ↪  c("l", "r"))
```

Property	Value
Vertex connectivity	0
Edge connectivity	0
Number of components	117
Size of largest component	1137
Number of isolates (degree 0)	78

Now, since vertex connectivity and edge connectivity is 0, this means that this airport network is connected to every single other airport in some path. In a hub-and-spoke network, these values are often low because removing just a few critical hubs would make collapse the network.

Transitivity

Transitivity, also called the clustering coefficient, measures the tendency for triangles to form in the network. In an airport context, a triangle means that if airport A has direct flights to both B and C, then B and C also have a direct flight between them.

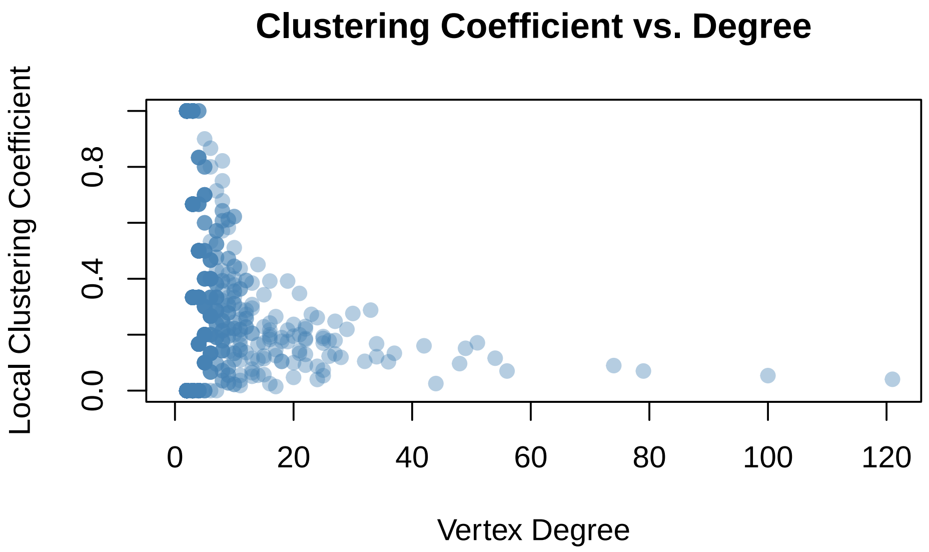
```
## Global transitivity
global_trans <- transitivity(flight_undirected, type = "global")

## Local transitivity
local_trans <- transitivity(flight_undirected, type = "local")

cat("Global transitivity (clustering coefficient):",
 ↪  round(global_trans, 4), "\n")
```

```
Global transitivity (clustering coefficient): 0.1402
```

```
cat("Average local transitivity:", round(mean(local_trans, na.rm
 ↪  = TRUE), 4), "\n")
```

```
Average local transitivity: 0.3549
```

```
## Local clustering vs degree
plot(igraph::degree(flight_undirected), local_trans,
     col = adjustcolor("steelblue", alpha.f = 0.4),
     pch = 19,
     xlab = "Vertex Degree",
     ylab = "Local Clustering Coefficient",
     main = "Clustering Coefficient vs. Degree")
```



Centrality Analysis

Centrality measures identify the most important or influential nodes in a network. I computed four classic centrality measures, each capturing a different notion of importance in th network.

Degree Centrality

Degree centrality tells us the number of direct connections a node has. In an airport network, this tells us which airports serve the most direct routes.

```
## Degree centrality
deg_cent <- igraph::degree(flight_undirected)

## Top 10 by degree
top_degree <- sort(deg_cent, decreasing = TRUE)[1:10]
kable(data.frame(Airport = names(top_degree),
                 Degree = as.integer(top_degree)),
      col.names = c("Airport (ICAO)", "Degree"),
      align = c("l", "r"),
      caption = "Top 15 Airports by Degree Centrality")
```

Table 8: Top 15 Airports by Degree Centrality

Airport (ICAO)	Degree
KDFW	121
KCLT	100
KIAD	79
KDAL	74
KFTW	56
KDTW	54
KTPA	51
KLAS	49
KTUL	48
KFCM	44

In the airport network, we chose Dallas-Forth Worth, Charlotte, Washington Dulles has some of the highest degree centrality meaning that they are the major hubs in the network.

Closeness Centrality

Closeness centrality measures how close a node is to all other nodes, computed as the inverse of the average shortest path distance. Airports with high closeness are well-positioned to reach the entire network quickly — they are geographically or topologically central.

```
## Closeness centrality on largest component
lcc <- induced_subgraph(flight_undirected,
                        which(comp$membership ==
                         ↪  which.max(comp$csize)))
close_cent <- igraph::closeness(lcc)

top_closeness <- sort(close_cent, decreasing = TRUE)[1:10]
kable(data.frame(Airport = names(top_closeness),
                 Closeness = round(as.numeric(top_closeness), 6)),
      col.names = c("Airport (ICAO)", "Closeness"),
      align = c("l", "r"),
      caption = "Top 15 Airports by Closeness Centrality")
```

Table 9: Top 15 Airports by Closeness Centrality

Airport (ICAO)	Closeness
KDFW	0.000213
KIAD	0.000206
KDAL	0.000202

Airport (ICAO)	Closeness
KTPA	0.000195
KHWO	0.000195
EGSS	0.000195
KCLT	0.000195
KFTW	0.000195
6FD7	0.000195
KRVS	0.000194

As expected, Dallas, Washington Dulles are high up on the list. But interestingly, this time Tampa International much higher than Charlotte. I think the most likely reason is that the Charlotte Airport has flights that end in the network and follow to more destinations. Tampa is in Florida, and those flights are much more connected inside of Florida and the southeast region, which makes it more central in terms of closeness.

Betweenness Centrality

Betweenness centrality counts the number of shortest paths between other pairs of nodes that pass through a given node. Airports with high betweenness are critical choke points.

```r
betw_cent <- igraph::betweenness(flight_undirected, normalized =
 ↪   TRUE)

top_betweenness <- sort(betw_cent, decreasing = TRUE)[1:10]
kable(data.frame(Airport = names(top_betweenness),
                 Betweenness = round(as.numeric(top_betweenness),
                  ↪   6)),
      col.names = c("Airport (ICAO)", "Betweenness"),
      align = c("l", "r"),
      caption = "Top 10 Airports by Betweenness Centrality")
```

Table 10: Top 10 Airports by Betweenness Centrality

Airport (ICAO)	Betweenness
KDFW	0.135991
KIAD	0.095267
KDAL	0.078444
KFTW	0.062871
KCLT	0.059499
EGSS	0.057008
KFCM	0.048783
KJFK	0.047787

Airport (ICAO)	Betweenness
KTUL	0.044639
KTPA	0.042151

Major hubs are again on top meaning that they connect flights from one airport to the other. For example, when I fly to San Francisco, I always have to take a layover at Chicago O'Hare which is a major hub in airport network in the United States.

Eigenvector Centrality

Eigenvector centrality extends the idea of degree centrality by weighting connections: being connected to well-connected airports matters more than being connected to poorly-connected ones. This captures the recursive notion that an airport is important if it is connected to other important airports.

```r
eig_cent <- eigen_centrality(flight_undirected)$vector

top_eigen <- sort(eig_cent, decreasing = TRUE)[1:10]
kable(data.frame(Airport = names(top_eigen),
                 Eigenvector = round(as.numeric(top_eigen), 6)),
      col.names = c("Airport (ICAO)", "Eigenvector Centrality"),
      align = c("l", "r"),
      caption = "Top 10 Airports by Eigenvector Centrality")
```

Table 11: Top 10 Airports by Eigenvector Centrality

Airport (ICAO)	Eigenvector Centrality
KDFW	1.000000
KDTW	0.728641
KCLT	0.697133
KLAS	0.681991
KIAD	0.531894
KORF	0.382264
KTUL	0.364943
KCLE	0.293416
KMKE	0.291355
KDAL	0.285431

Hub and Authority Scores

For directed networks,hub and authority scores provide a complementary perspective. An airport is a good **hub** if it sends flights to many good authorities, and a good **authority** if it receives flights from many good hubs. In aviation, hubs are airports that serve as major departure points and authorities are major arrival destinations.

```r
hub_scores <- hub_score(flight_network)$vector
auth_scores <- authority_score(flight_network)$vector

top_hubs <- sort(hub_scores, decreasing = TRUE)[1:10]
top_auths <- sort(auth_scores, decreasing = TRUE)[1:10]

kable(data.frame(Hub_Airport = names(top_hubs),
                 Hub_Score = round(as.numeric(top_hubs), 4),
                 Auth_Airport = names(top_auths),
                 Auth_Score = round(as.numeric(top_auths), 4)),
      col.names = c("Hub Airport", "Hub Score", "Authority
      ↪  Airport", "Authority Score"),
      align = c("l", "r", "l", "r"),
      caption = "Top 10 Airports by Hub and Authority Scores")
```

Table 12: Top 10 Airports by Hub and Authority Scores

Hub Airport	Hub Score	Authority Airport	Authority Score
KDFW	1.0000	KDFW	1.0000
KDTW	0.9011	KDTW	0.5967
KCLT	0.8776	KLAS	0.5553
KLAS	0.8467	KCLT	0.5549
KIAD	0.6419	KIAD	0.4430
KTPA	0.4864	KORF	0.3059
KTUL	0.4628	KU42	0.2988
KORF	0.4533	KDAL	0.2758
KCLE	0.4341	KTUL	0.2757
KMKE	0.4055	KJFK	0.2474

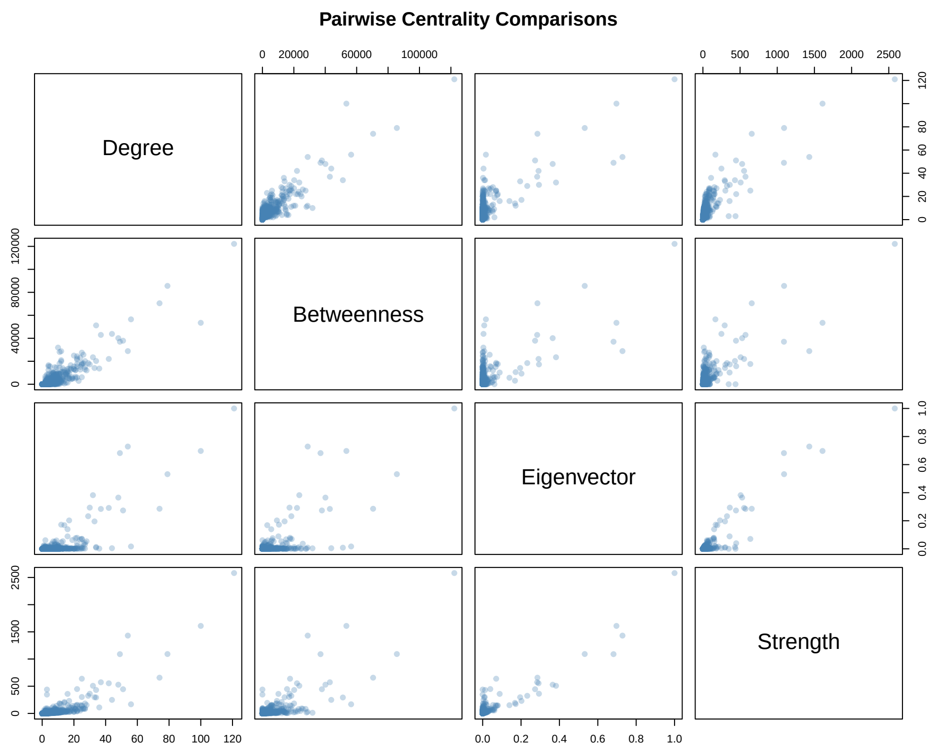
I was surprised that Detroit Wayne County Airport wasn't high up there in the hub scores because Detroit is a major Delta hub. One of the reasons that I could think of that is Detroit is not a well sought-after destination for travel compared to Chicago, Dallas, San Francisco, or New York. But it seems like Detroit is a good location for making connecting hub.

Comparing Centrality Measures

Different centrality measures capture different aspects of importance. I wanted to see how correlated they are in this airport network.

```
## Pairwise centrality comparison
```

```
cent_df <- data.frame(
  airport = V(flight_undirected)$name,
  degree = igraph::degree(flight_undirected),
  betweenness = igraph::betweenness(flight_undirected),
  eigenvector =
   ↪  igraph::eigen_centrality(flight_undirected)$vector,
  strength = igraph::strength(flight_undirected)
)

## Pairwise scatter plots
pairs(cent_df[, c("degree", "betweenness", "eigenvector",
 ↪  "strength")],
      col = adjustcolor("steelblue", alpha.f = 0.3),
      pch = 19,
      main = "Pairwise Centrality Comparisons",
      labels = c("Degree", "Betweenness", "Eigenvector",
       ↪  "Strength"))
```



Now, here all the centrality measures are positively correlated meaning that the major hubs tend to score high across all measures. However, the strength (weighted degree) shows a stronger correlation with eigenvector centrality than with betweenness, which makes sense because both strength and eigenvector centrality capture the idea of being connected to other important nodes. Betweenness can be high for airports that serve as critical bridges even if they don't have many direct connections.