Network auto-correlation for desire lines between tracts in Brooklyn

The anaylsis paralles that of the NTA auto-correlation. However, it is at the tract resolution, instead of the NTA resolution. Initially, I attempted to examine the auto-correlation for all tracts in Brooklyn. However, there are 133000 unique trips between tracts in Brooklyn. Checking whether an undirected edge between each tract pair is 133000 choose 2, or close to 9billion operations. This overwhelmed my computer. Instead, I arbitrarily picked an NTA and examined the trips between tracts within that NTA. I repeated this a few times with different NTAs. Only one analysis shown in these results. But, it would be straight-forward to adjust the code to iterate through the NTAs and record the results.

```
library(tidyverse)
library(tidygraph)
library(ggraph)
library(spData)
library(spdep)
library(igraph)
options(scipen = 999)
```

Define each node as a trip

```
bk_name <- "Brooklyn"
bk_county_code <- "047"

bk_ods <- readr::read_csv('./data/ny_od_main_JT00_2019.csv') %>%
  dplyr::filter(
    stringr::str_sub(as.character(w_geocode), 3, 5) == bk_county_code &
    stringr::str_sub(as.character(h_geocode), 3, 5) == bk_county_code
  )
```

Note Attempt to use all census tracts in borough overwhelmed my cpu Code related to analyzing the whole borough is demoted to read-only

```
trip_nodes <- bk_ods %>%
  dplyr::mutate(
    w_tract = stringr::str_sub(as.character(w_geocode), 6, 11),
    h_tract = stringr::str_sub(as.character(h_geocode), 6, 11),
  ) %>%
  dplyr::filter(w_tract != h_tract) %>%
  dplyr::mutate(
    trip = stringr::str_c(
      "CT",
      ifelse(w_tract < h_tract, w_tract, h_tract),
      "CT",
      ifelse(w_tract > h_tract, w_tract, h_tract)
      )
  ) %>%
  dplyr::select(trip, S000) %>%
  dplyr::group_by(trip) %>%
  dplyr::summarise(
    S000 = sum(S000)
  ) %>%
  unique()
```

Define each edge as a shared destination between two trips

```
build_edges <- function(nodes){
  edges_from <- vector()
  edges_to <- vector()
  nodes_count <- length(nodes)
  for(i in 1:(nodes_count - 1)) {
    offset <- i + 1
    from_node <- nodes[i]
    from_one <- stringr::str_sub(from_node, 1, 8)
    from_two <- stringr::str_sub(from_node, 9, 16)
    for(j in offset:nodes_count){
      to_node <- nodes[j]
      are_neighbors <- stringr::str_detect(to_node, from_one) | stringr::str_detect(to_node, from_two)
      if (are_neighbors) {
        edges_from <- append(edges_from, from_node)
        edges_to <- append(edges_to, to_node)
      }
    }
  }
  return (tibble::tibble(from = edges_from, to = edges_to))
}
```

```
trip_edges <- build_edges(trip_nodes$trip)

trip_network <- tidygraph::tbl_graph(
  nodes = trip_nodes,
  edges = trip_edges
  )
```

Network auto-correlation of desire lines between census tracts within an NTA

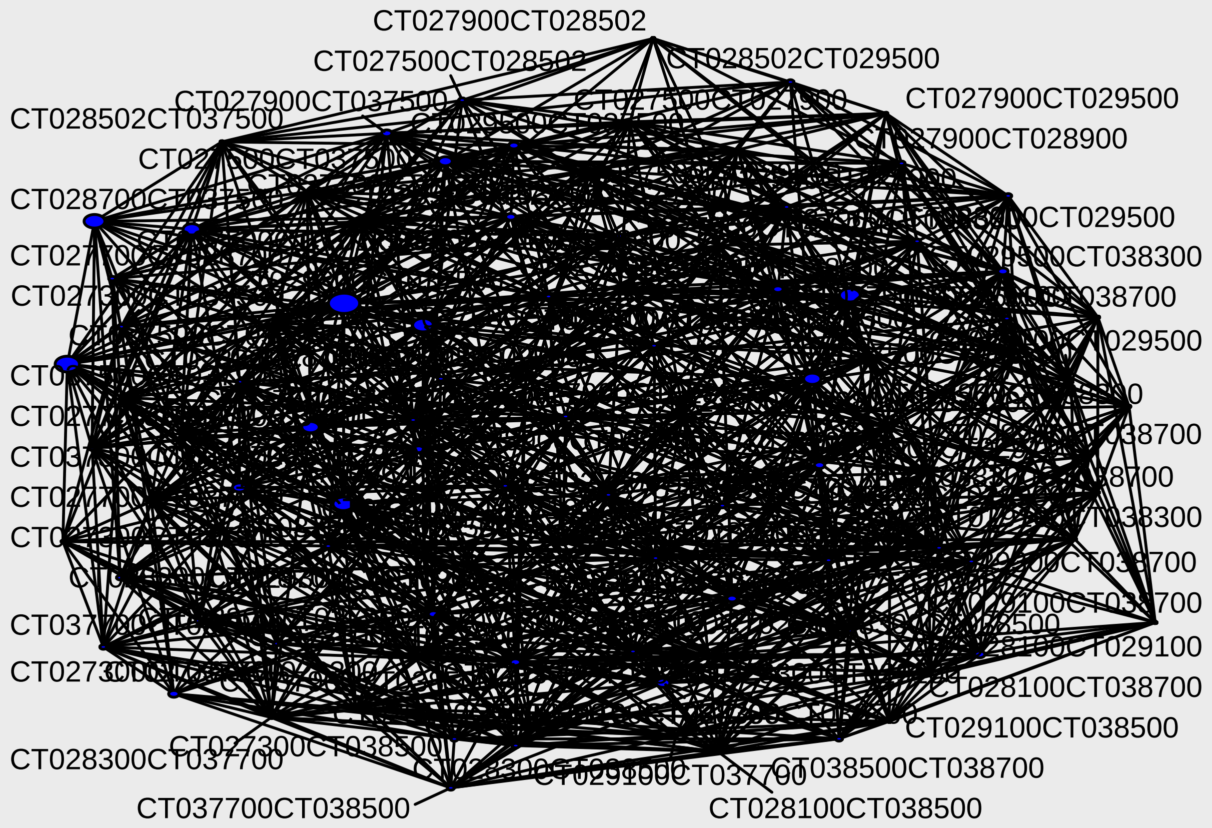
```
tract_nta_equiv <- readxl::read_xlsx('./data/nyc_2010_census_tract_nta_equiv.xlsx')
bk_tract_nta_equiv <- tract_nta_equiv %>%
  dplyr::filter(borough_name == bk_name)

bk_nta_codes <- unique(bk_tract_nta_equiv$nta_code)

## Used the first number that popped into my head- psuedo random but repeatable
nta_of_interest <- bk_nta_codes[46]
tracts_oi <- bk_tract_nta_equiv %>%
  dplyr::filter(nta_code == nta_of_interest)
tracts_oi <- tracts_oi$census_tract

trips_nodes_oi <- bk_ods %>%
  dplyr::mutate(
    w_tract = stringr::str_sub(as.character(w_geocode), 6, 11),
    h_tract = stringr::str_sub(as.character(h_geocode), 6, 11),
  ) %>%
  dplyr::filter(w_tract != h_tract) %>%
  dplyr::filter(w_tract %in% tracts_oi & h_tract %in% tracts_oi) %>%
  dplyr::mutate(
    trip = stringr::str_c(
      "CT",
      ifelse(w_tract < h_tract, w_tract, h_tract),
      "CT",
      ifelse(w_tract > h_tract, w_tract, h_tract)
      )
  ) %>%
  dplyr::select(trip, S000) %>%
  dplyr::group_by(trip) %>%
  dplyr::summarise(
    S000 = sum(S000)
  ) %>%
  unique()
```

```
trip_edges_oi <- build_edges(trips_nodes_oi$trip)

trip_network_oi <- tidygraph::tbl_graph(
  nodes = trips_nodes_oi,
  edges = trip_edges_oi,
  node_key = "trip"
  )
```

```
total_trips_oi <- sum(trips_nodes_oi$S000)

ggraph::ggraph(trip_network_oi, layout="stress") +
  ggraph::geom_edge_link() +
  ggraph::geom_node_circle(aes(r = (trips_nodes_oi$S000 / total_trips_oi)), fill="blue") +
  ggraph::geom_node_text(aes(label = trips_nodes_oi$trip), repel = TRUE)
```

```
## Warning: Using the `size` aesthetic in this geom was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` in the `default_aes` field and elsewhere instead.
```

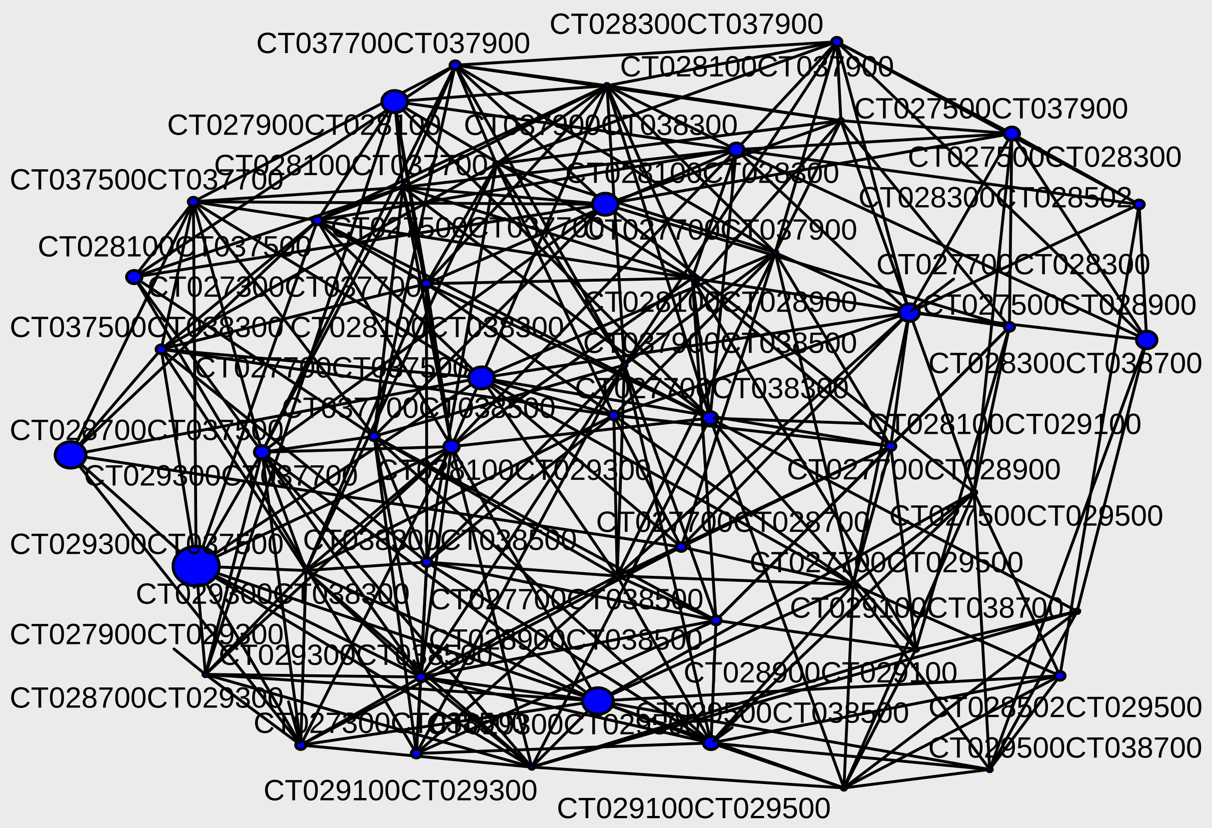
```
## Warning: ggrepel: 1 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



## Subgraph Use a sub section of the network to make it easier to visualize

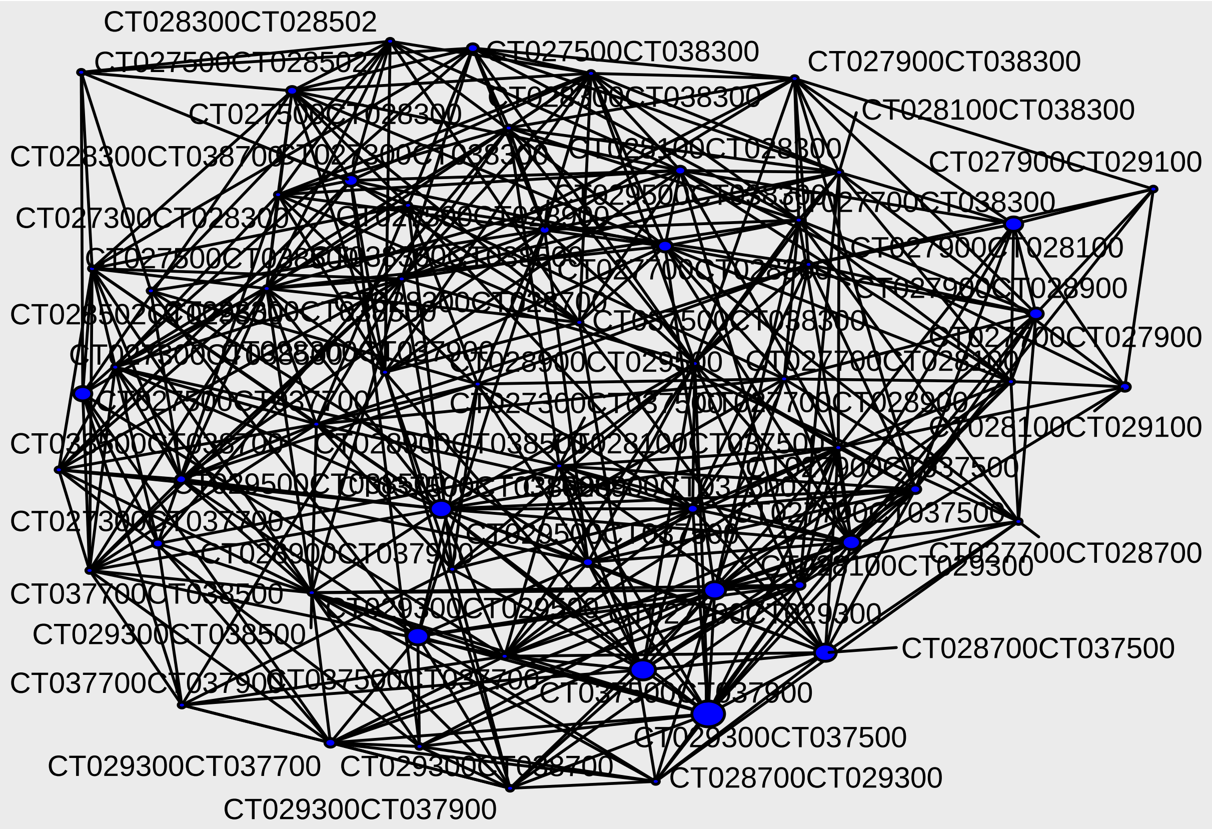
Random sample

```
trip_nodes_rand <- trips_nodes_oi %>%
  dplyr::slice_sample(n = 50)
trip_edges_rand <- build_edges(trip_nodes_rand$trip)
trip_network_rand <- tidygraph::tbl_graph(
  nodes = trip_nodes_rand,
  edges = trip_edges_rand,
  node_key = "trip"
  )

total_trips_rand <- sum(trip_nodes_rand$S000)
ggraph::ggraph(trip_network_rand, layout="stress") +
  geom_edge_link() +
  geom_node_circle(aes(r = (trip_nodes_rand$S000 / total_trips_rand)), fill = "blue") +
  geom_node_text(aes(label = trip_nodes_rand$trip), repel = TRUE)
```



Most popular trips

```
trip_nodes_rand <- trips_nodes_oi %>%
  dplyr::slice_max(order_by = S000, n = 50)
trip_edges_rand <- build_edges(trip_nodes_rand$trip)
trip_network_rand <- tidygraph::tbl_graph(
  nodes = trip_nodes_rand,
  edges = trip_edges_rand,
  node_key = "trip"
  )

total_trips_rand <- sum(trip_nodes_rand$S000)
ggraph::ggraph(trip_network_rand, layout="stress") +
  geom_edge_link() +
  geom_node_circle(aes(r = (trip_nodes_rand$S000 / total_trips_rand)), fill = "blue") +
  geom_node_text(aes(label = trip_nodes_rand$trip), repel = TRUE)
```



Global Moran's I

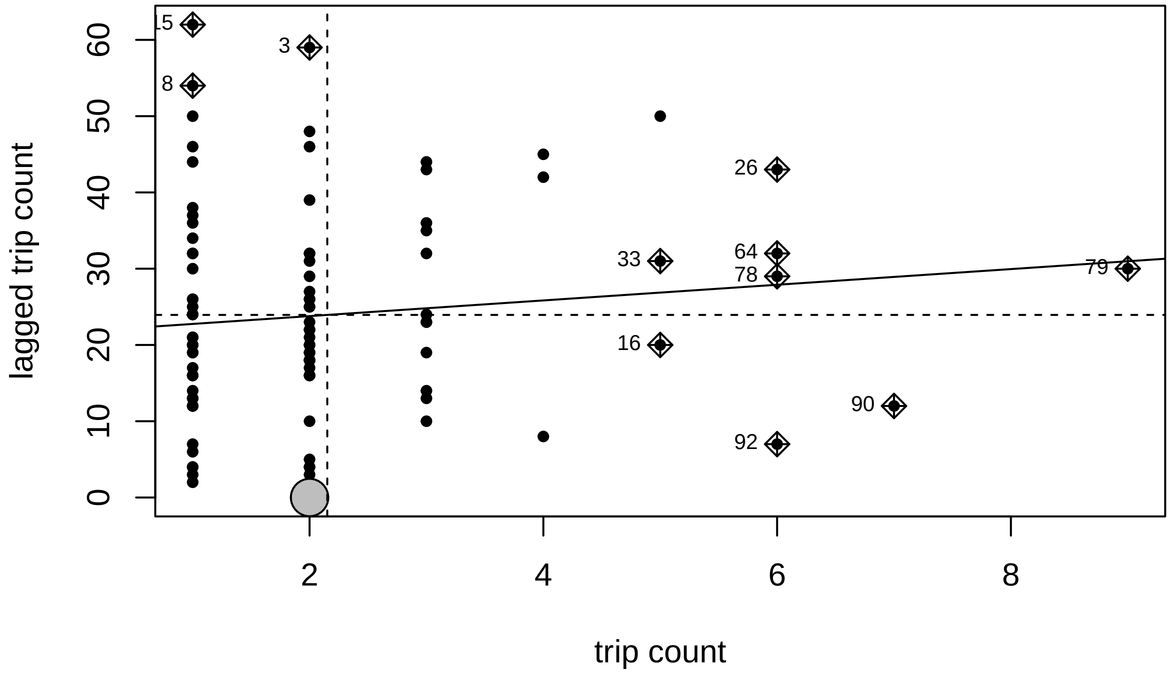
```
trip_network_weights <- trip_network_oi %>%
  igraph::as_adj() %>%
  spdep::mat2listw()
```

```
## Warning in sn2listw(df): 99 is not an origin
```

```
global_morans <- spdep::moran.test(trips_nodes_oi$S000, trip_network_weights, zero.policy = TRUE)
global_morans
```

```
##
##  Moran I test under randomisation
##
## data:  trips_nodes_oi$S000
## weights: trip_network_weights  n reduced by no-neighbour observations
##
##
## Moran I statistic standard deviate = 3.0402, p-value = 0.001182
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      0.0691208501     -0.0103092784      0.0006825938
```

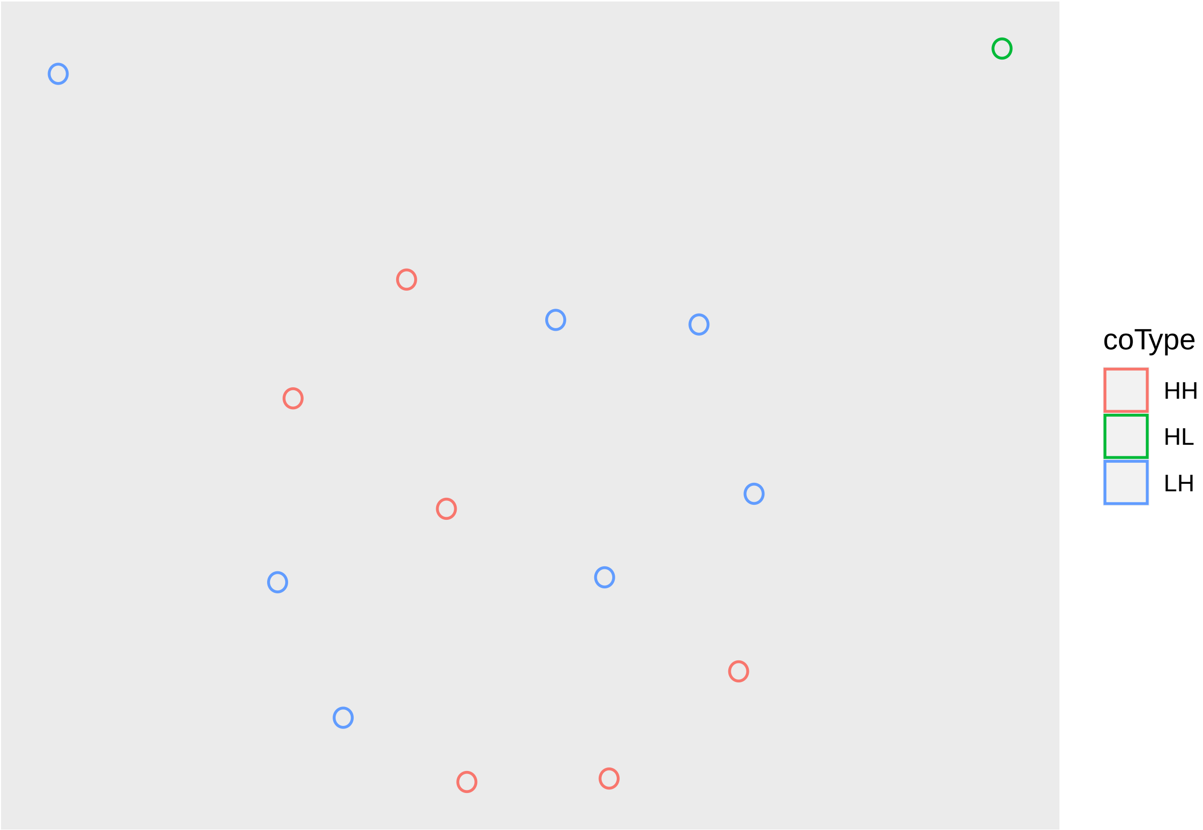
```
spdep::moran.plot(
  trips_nodes_oi$S000,
  trip_network_weights,
  zero.policy = TRUE,
  xlab = "trip count",
  ylab = "lagged trip count",
  pch = 20,
)
```



Local Indicators of spatial autocorrelation

```
local_moran <- spdep::localmoran(
  trips_nodes_oi$S000,
  trip_network_weights,
  zero.policy = TRUE,
  na.action = na.omit,
)

sig_lev <- 0.05
avg_trip_count <- mean(trips_nodes_oi$S000)

lisa_classes <- local_moran %>%
  tibble::as_tibble() %>%
  magrittr::set_colnames(
    c("Ii","E.Ii","Var.Ii","Z.Ii","Pr(z > 0)")
  ) %>%
  dplyr::mutate(
    coType = dplyr::case_when(
      `Pr(z > 0)` > 0.05 ~ "Insignificant",
      `Pr(z > 0)` <= 0.05 & Ii >= 0 & trips_nodes_oi$S000 >= avg_trip_count ~ "HH",
      `Pr(z > 0)` <= 0.05 & Ii >= 0 & trips_nodes_oi$S000 <  avg_trip_count ~ "LL",
      `Pr(z > 0)` <= 0.05 & Ii < 0 & trips_nodes_oi$S000 >= avg_trip_count ~ "HL",
      `Pr(z > 0)` <= 0.05 & Ii < 0 & trips_nodes_oi$S000 < avg_trip_count ~ "LH"
    )
  )
```

```
trip_network_cluster <- trip_network_oi %>%
  tidygraph::activate(nodes) %>%
  dplyr::mutate(coType = lisa_classes$coType %>% tidyr::replace_na("Insignificant"))

tract_sig <- trip_network_cluster %>%
  dplyr::filter(coType != "Insignificant")

ggraph::ggraph(tract_sig, layout="stress") +
  ggraph::geom_node_circle(aes(r = 0.025, color = coType))
```


coType
HH
HL
LH