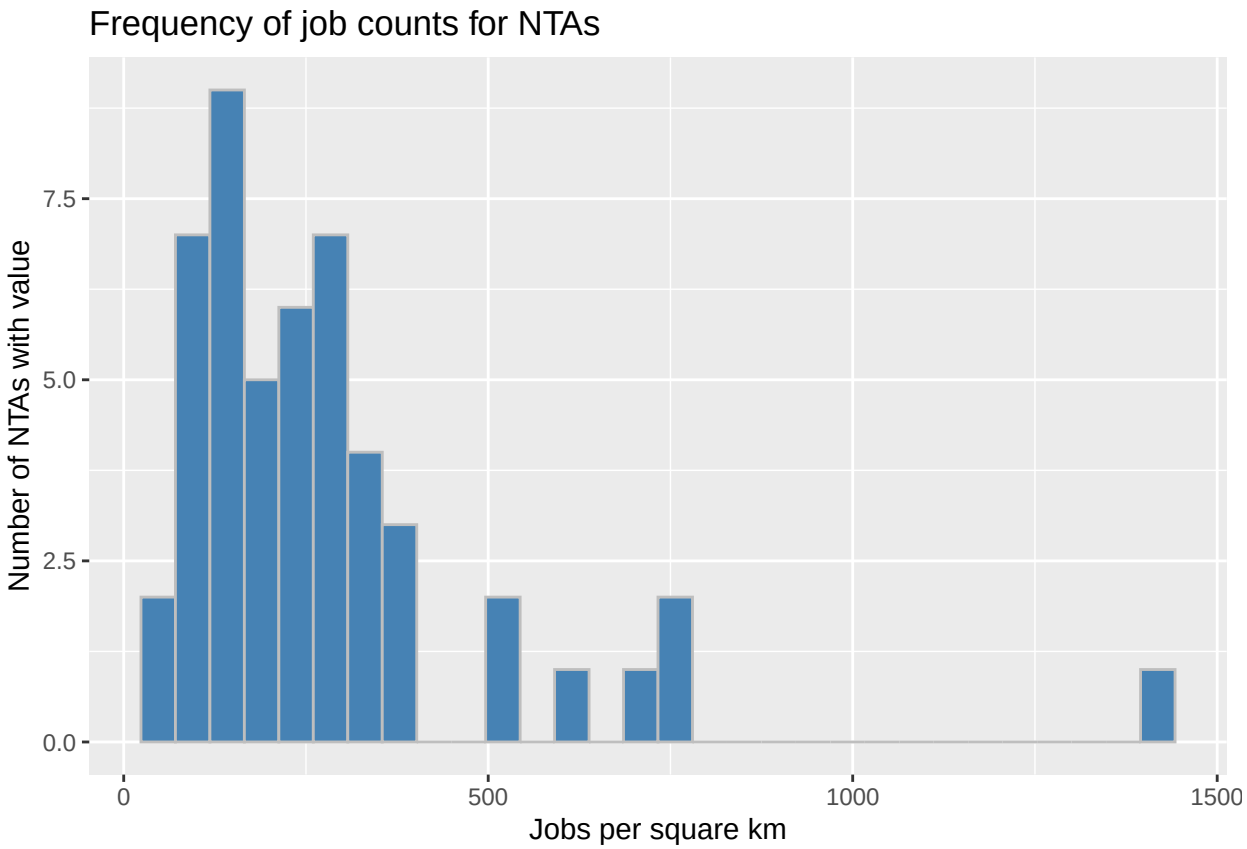


Figure 1: Frequency of job counts in each NTAs, standardized by the area of the NTA

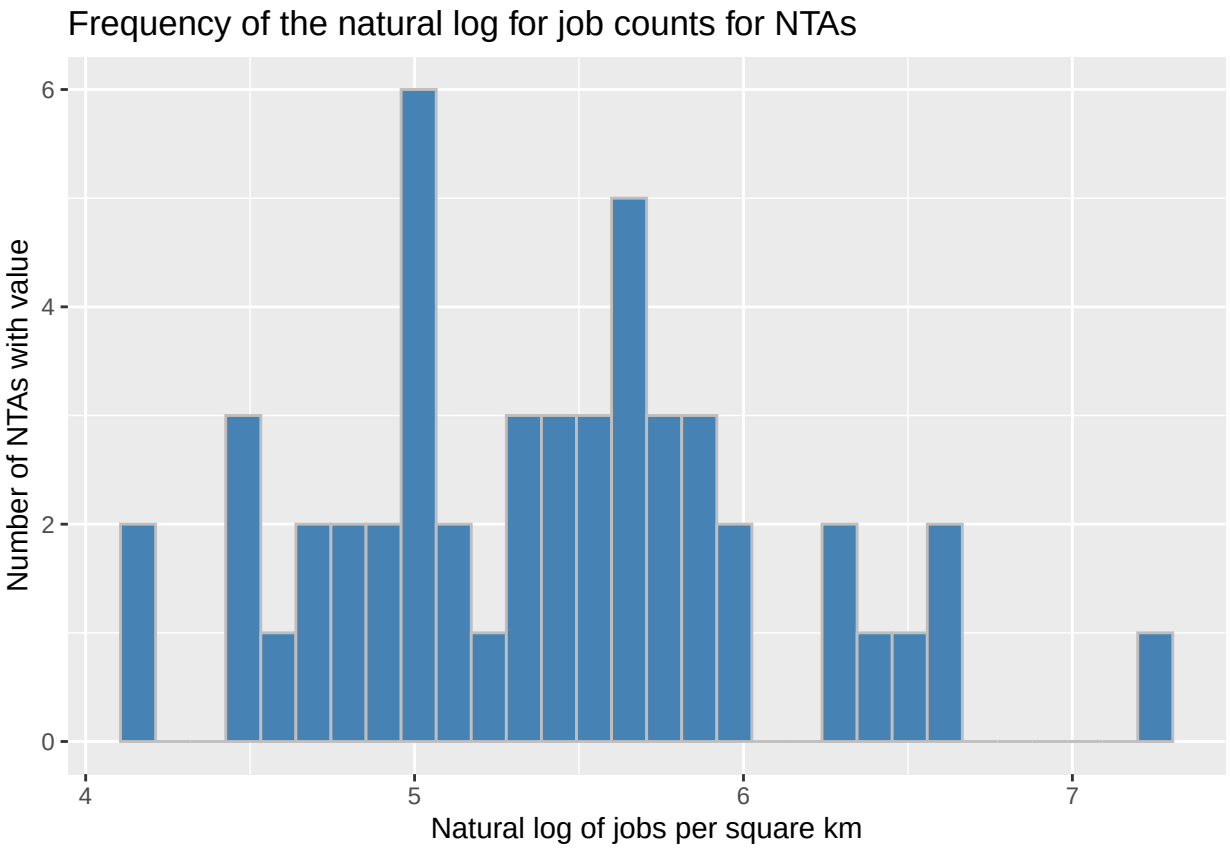


Figure 2: Frequency of natural log for job counts in each NTA, standardized by the area of the NTA

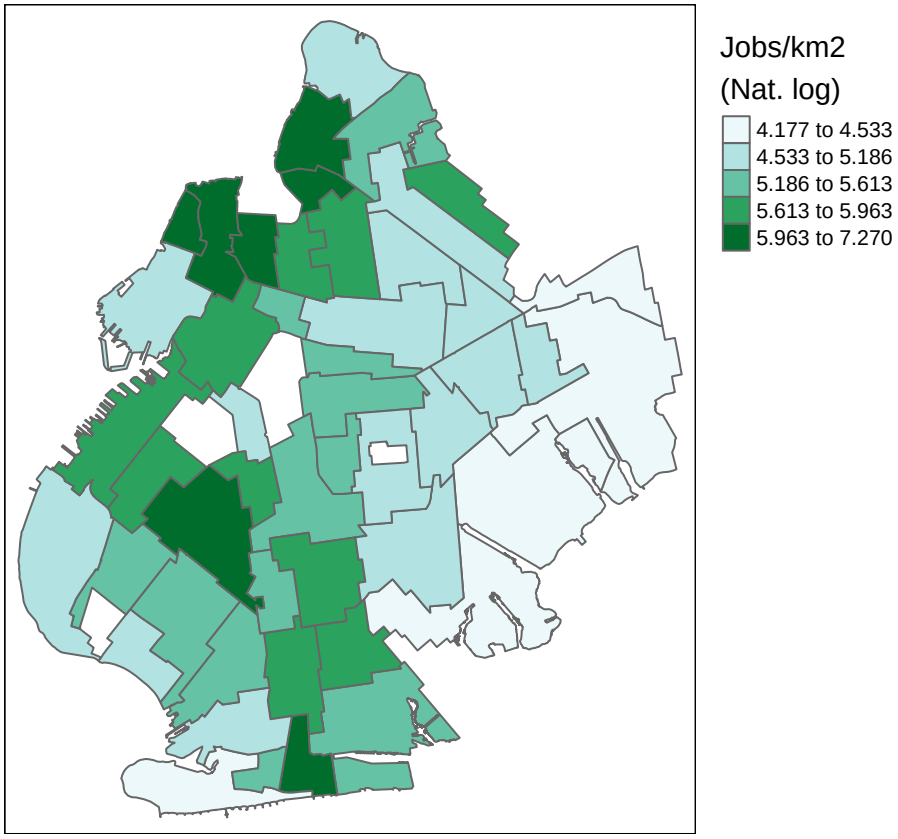


Figure 3: Number of jobs in each NTA tract. Job count is calculated from the sum of all trips which end in that NTA. It is standardized by the total area of the NTA. It is also transformed with a natural log.

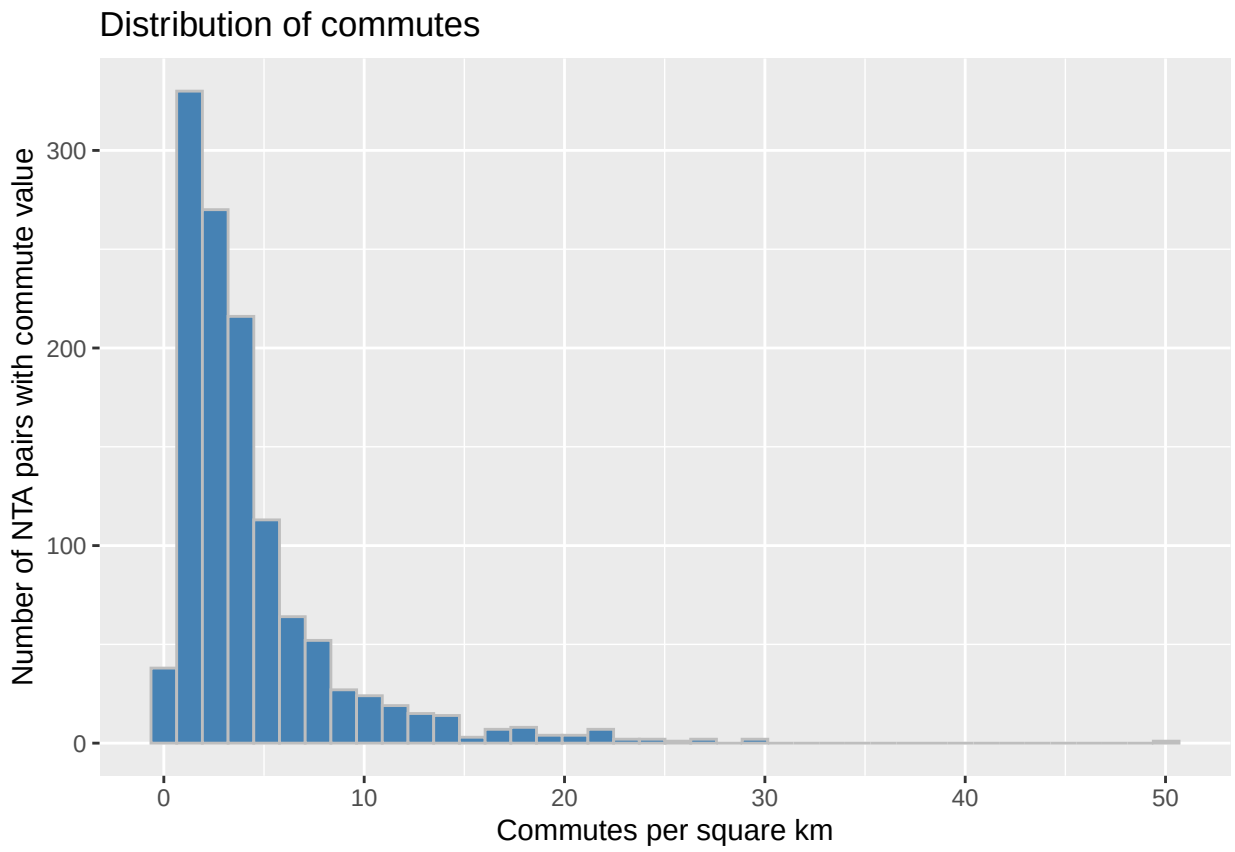


Figure 4: Frequency of commute counts, standardized by the sum of the areas of the commute's origin and destination NTAs

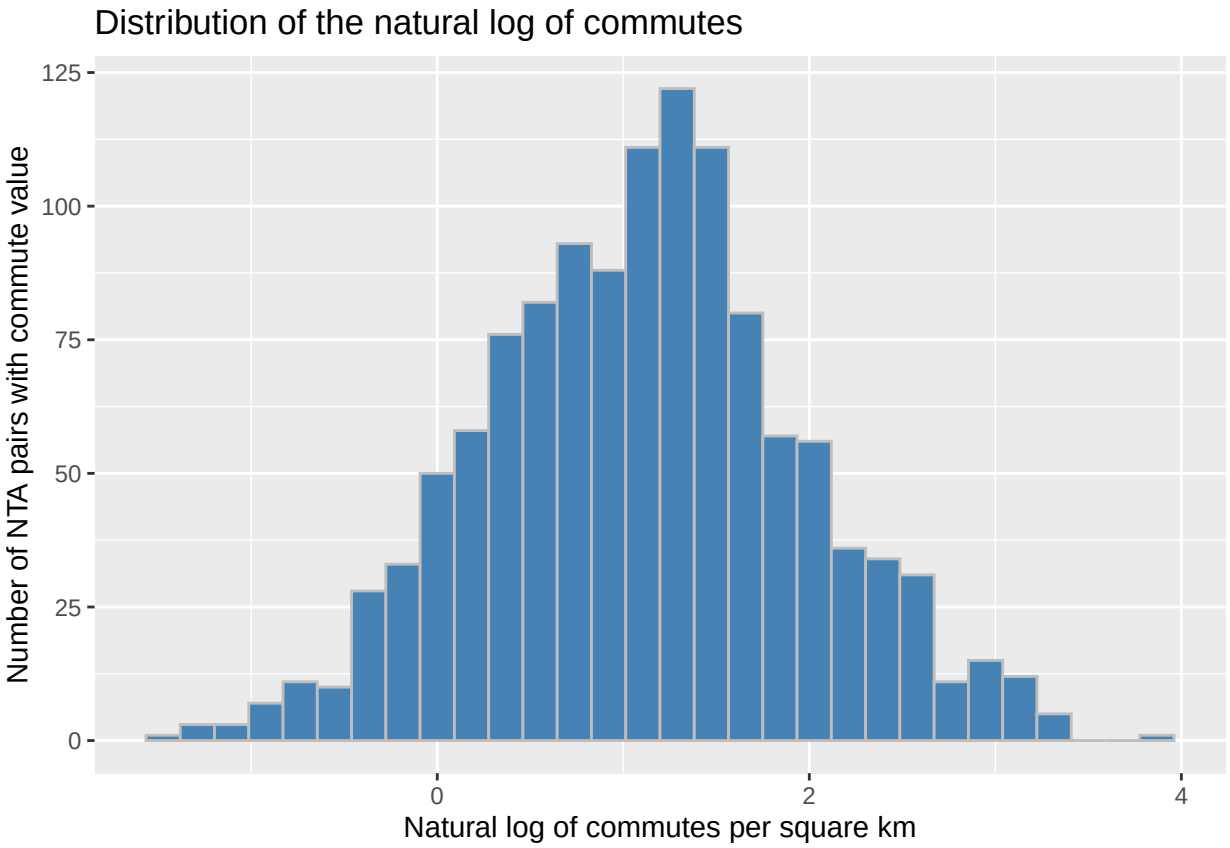


Figure 5: Frequency of the natural log of commute counts, standardized by the sum of the areas of the commute's origin and destination NTAs

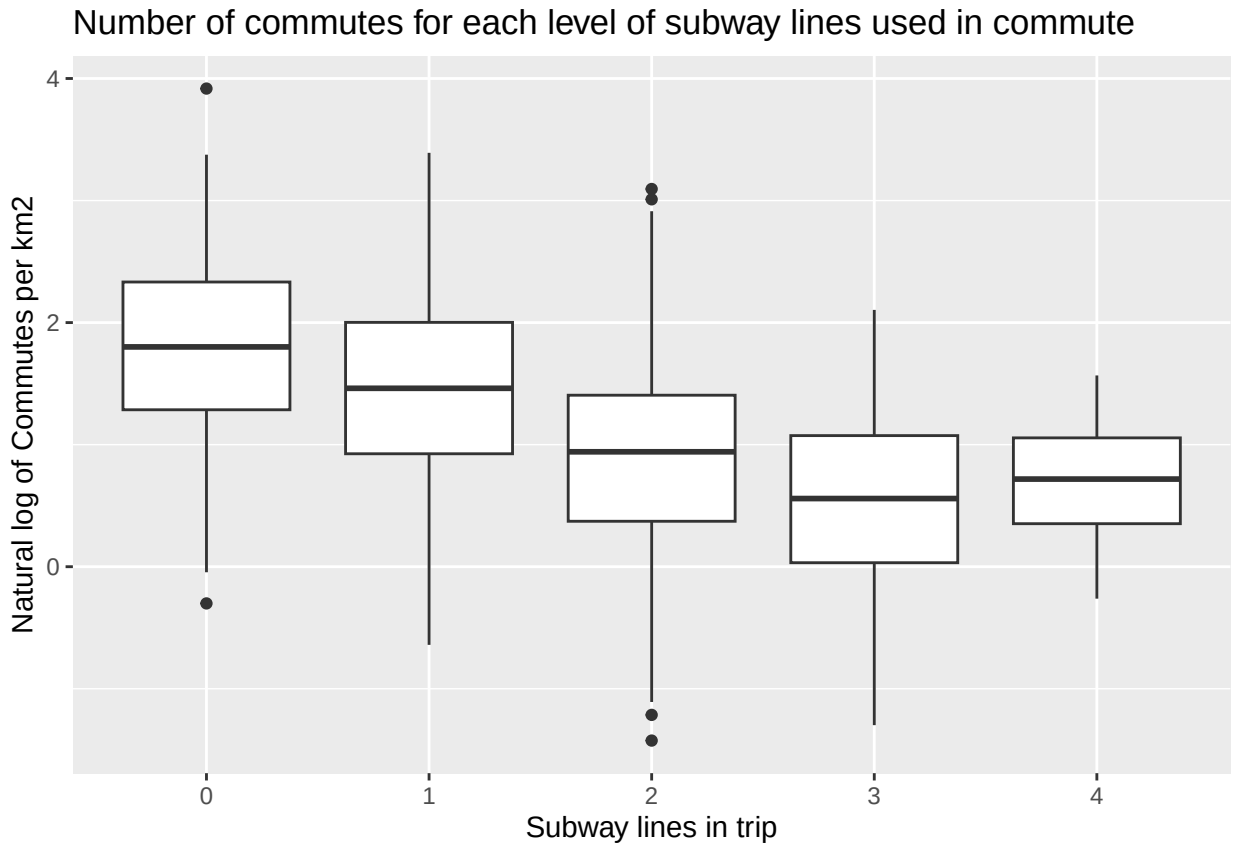


Figure 6: Boxplot for the number of commutes, factored by the number of subway lines used during the commute. Commutes are transformed by a natural log and standardized by area

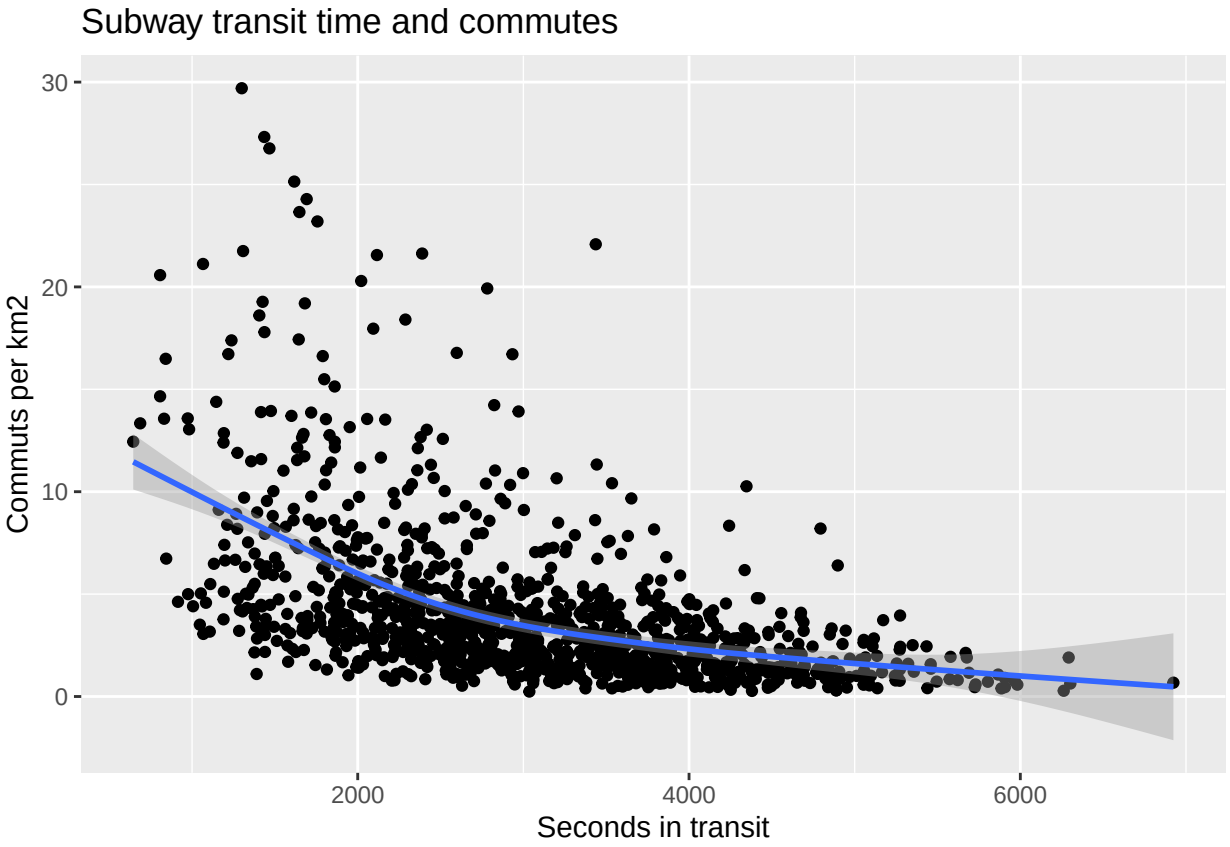


Figure 7: Time in transit on the subway plotted against the number of commutes standardized by area. A smoother line is overlayed in blue.

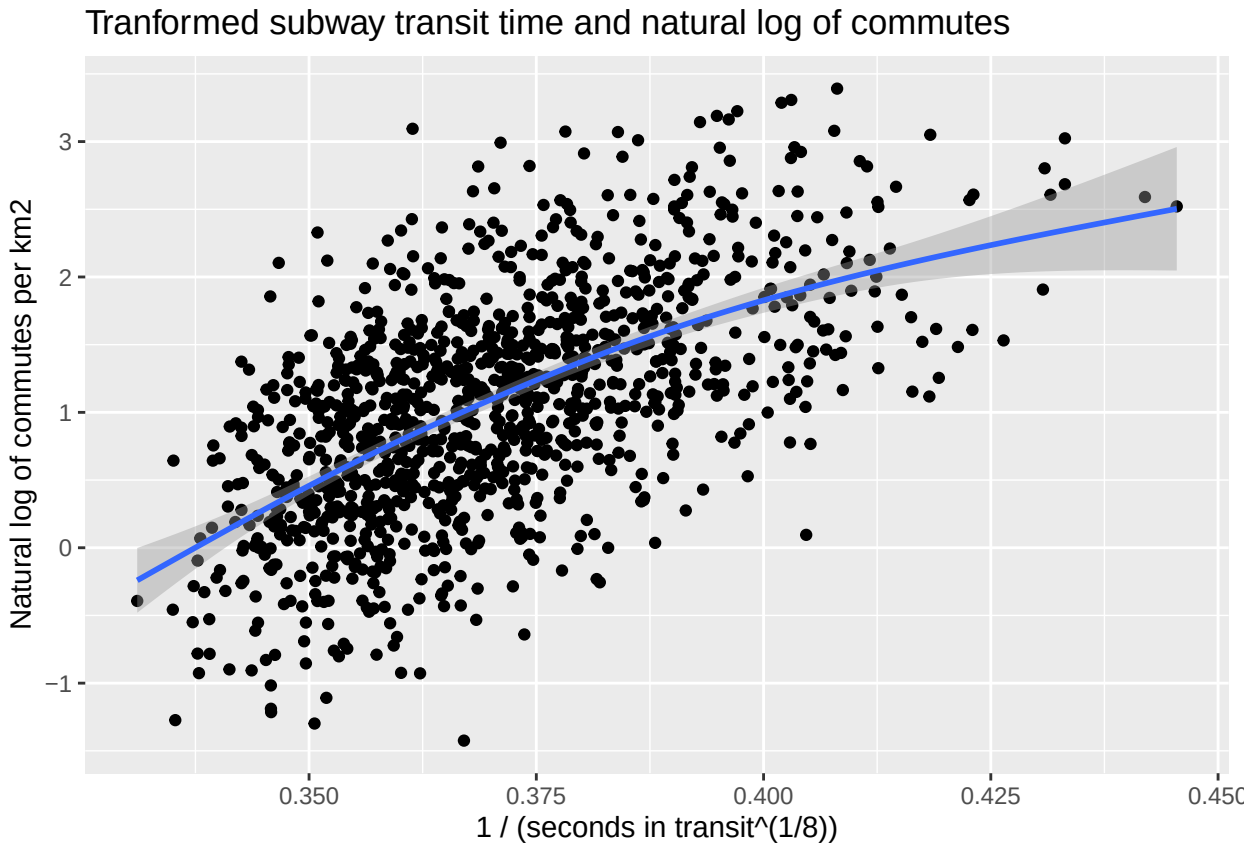


Figure 8: Time in subway transit is transormed by raising it to the power of one-eighth and taking the inverse. Commute count is transformed with the natural log. The values are plotted against each other. A smoother line is overlayed in blue.

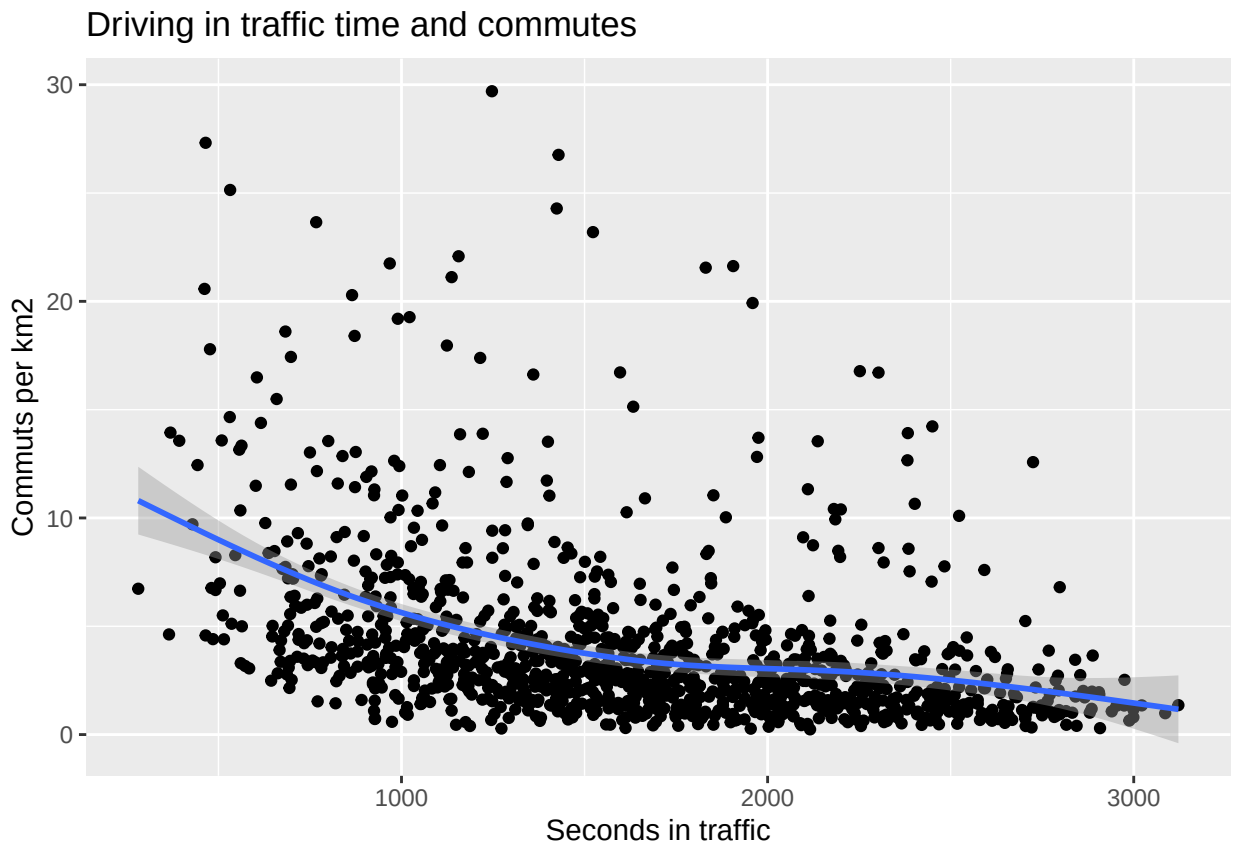
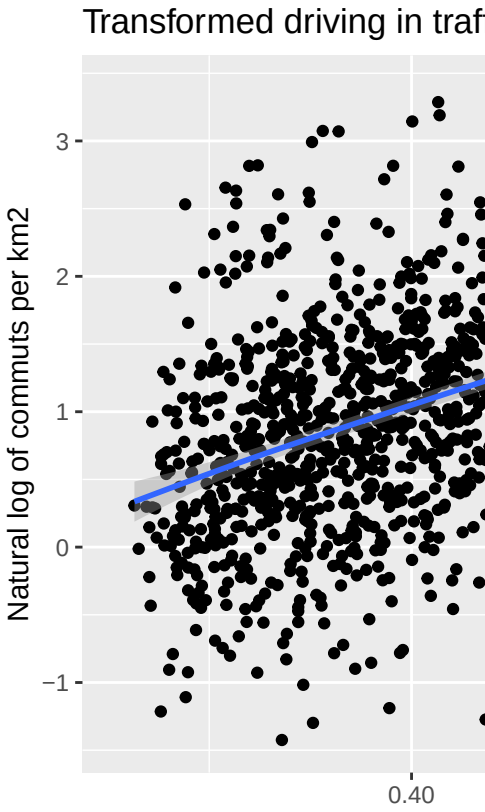


Figure 9: Time driving in traffic plotted against the number of commutes standardized by area. A smoother line is overlayed in blue.



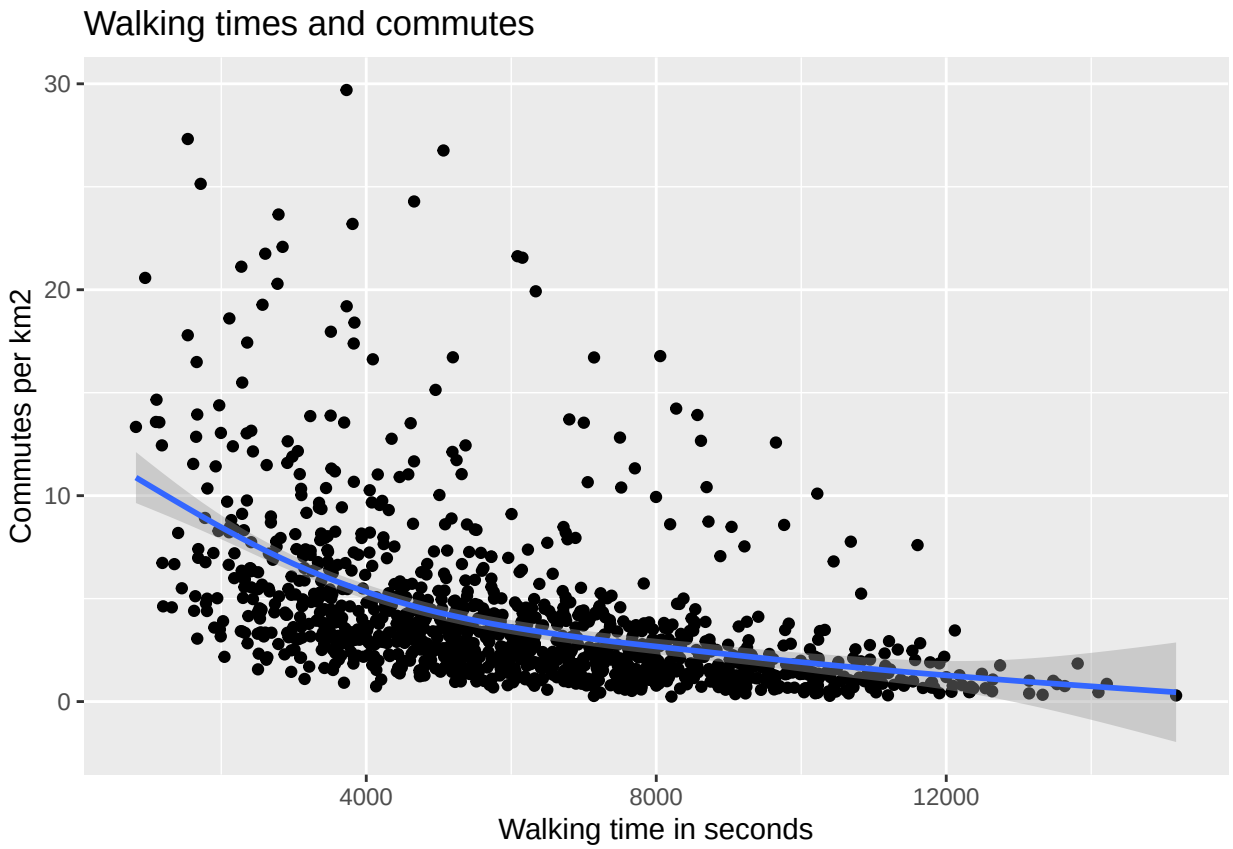
The same transformations for subway trip times are applied to driving in traffic times.

Walking time and commute count

We also remove the 77 points from the walking linear regression and apply the same transformations.

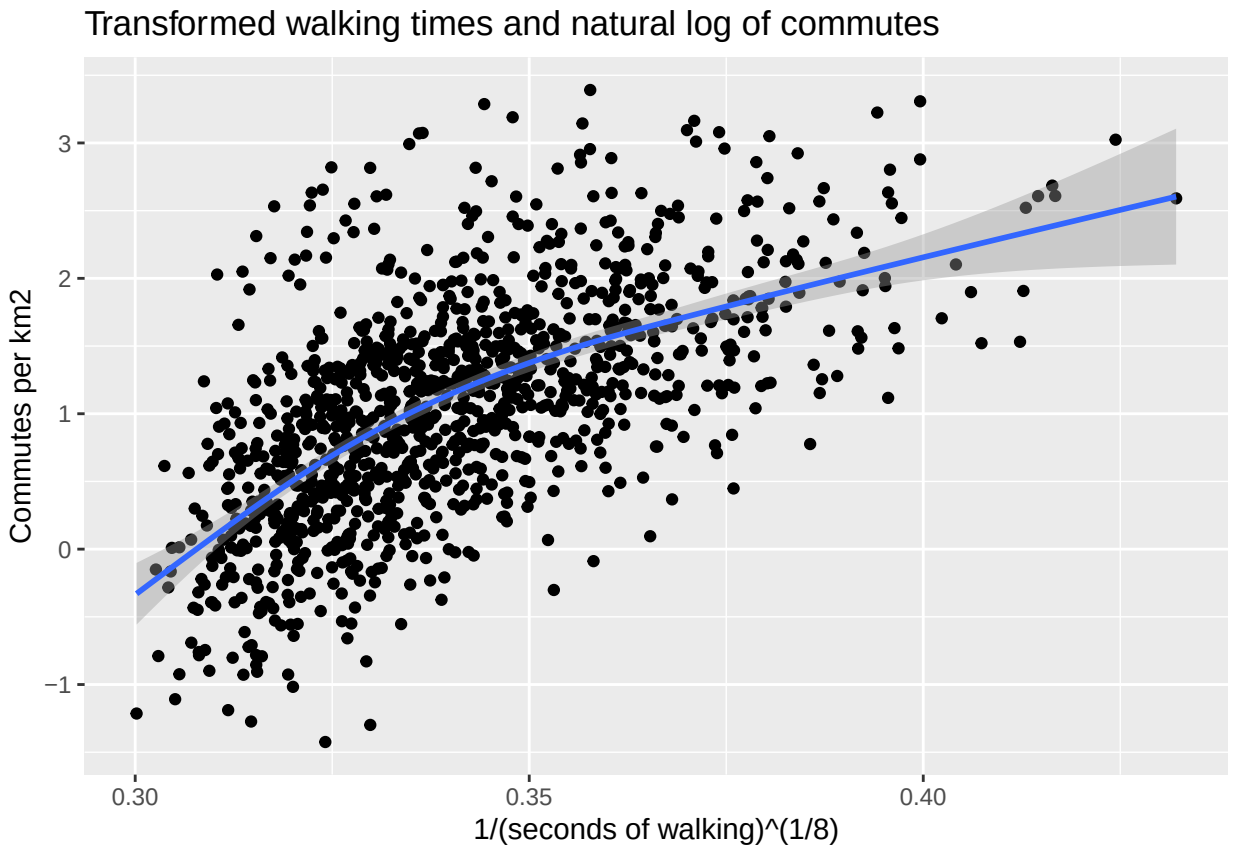
```
walking_times <- walking_times %>%
  dplyr::mutate(
    log_S000 = log(S000),
    S000_km2 = commute_counts$S000_km2,
    log_S000_km2 = commute_counts$log_S000_km2,
    i_seconds_of_walking = 1 / seconds_of_walking^(1/8)
  )

walking_times_reduced <- walking_times %>%
  dplyr::filter(
    trip %in% subway_times_connected$trip
  )
```

```
ggplot(data = walking_times_reduced, aes(x = seconds_of_walking, y = S000_km2)) +
  ggtitle("Walking times and commutes") +
  xlab("Walking time in seconds") +
  ylab("Commutes per km2") +
  geom_point() +
  stat_smooth()
```



Transformed

```
ggplot(data = walking_times_reduced, aes(x = i_seconds_of_walking, y = log_S000_km2)) +
  ggtitle("Transformed walking times and natural log of commutes") +
  xlab("1/(seconds of walking)^(1/8)") +
  ylab("Commutes per km2") +
  geom_point() +
  stat_smooth()
```



Regression of Subway, Driving, and walking

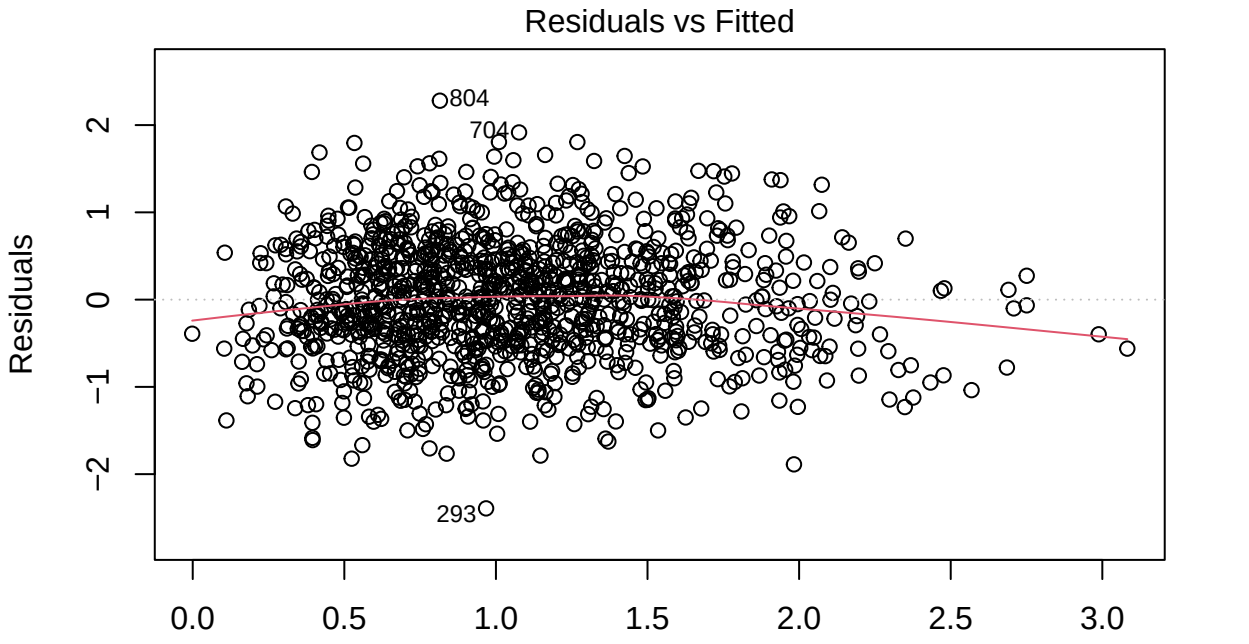
Subway model

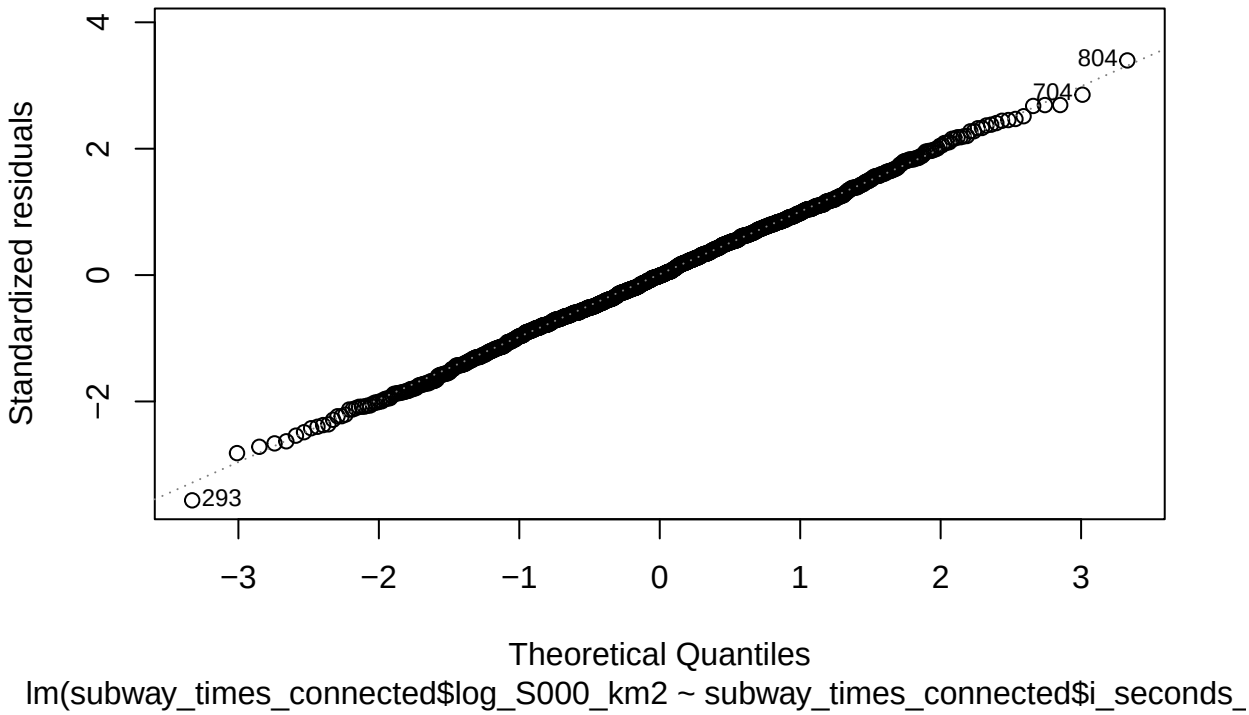
```
subway_connected_model <- lm(subway_times_connected$log_S000_km2 ~ subway_times_connected$i_seconds_in_
summary(subway_connected_model)
```

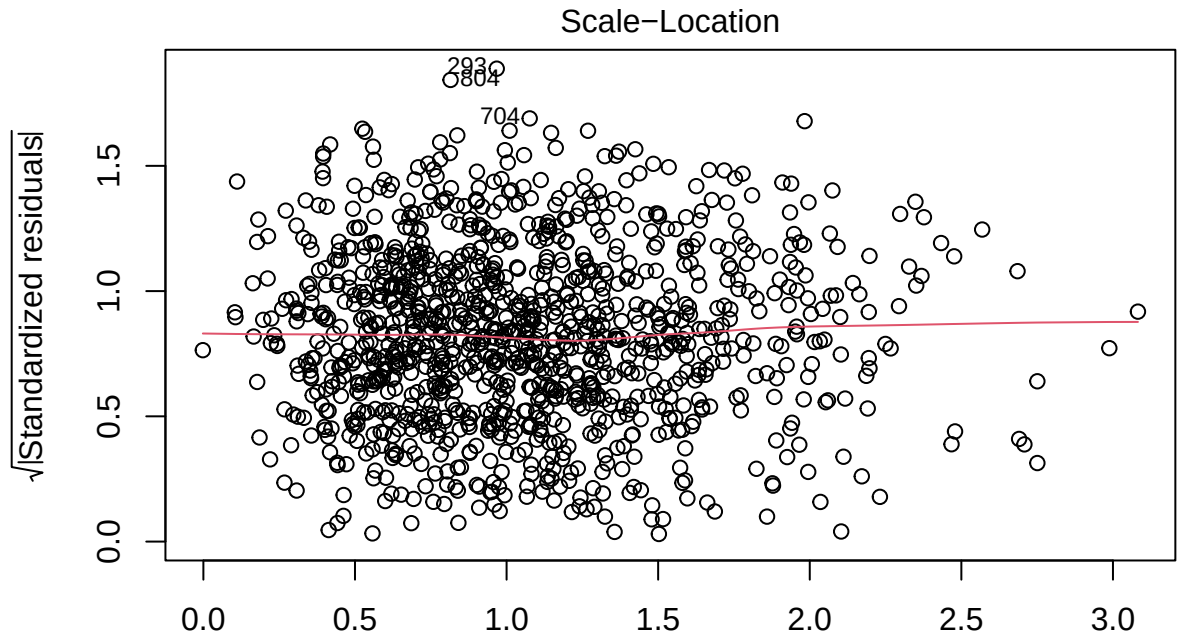
```
##
## Call:
## lm(formula = subway_times_connected$log_S000_km2 ~ subway_times_connected$i_seconds_in_transit)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.39218 -0.43877 -0.00603  0.45819  2.27949
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                  -8.9314     0.4023  -22.20
## subway_times_connected$i_seconds_in_transit  26.9699     1.0849   24.86
##                                                    Pr(>|t|)
## (Intercept)                                <0.0000000000000002 ***
## subway_times_connected$i_seconds_in_transit <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6714 on 1146 degrees of freedom
## Multiple R-squared:  0.3503, Adjusted R-squared:  0.3498
```

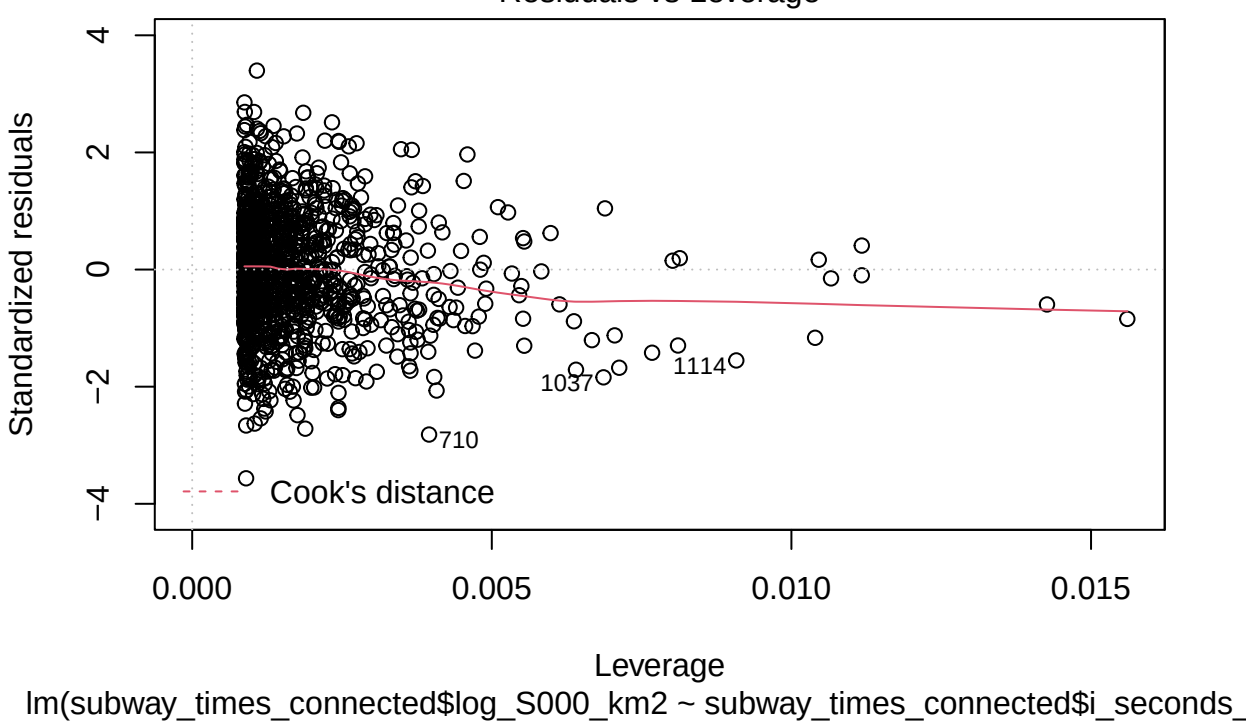
```
## F-statistic:   618 on 1 and 1146 DF,  p-value: < 0.00000000000000022
```

```
plot(subway_connected_model)
```









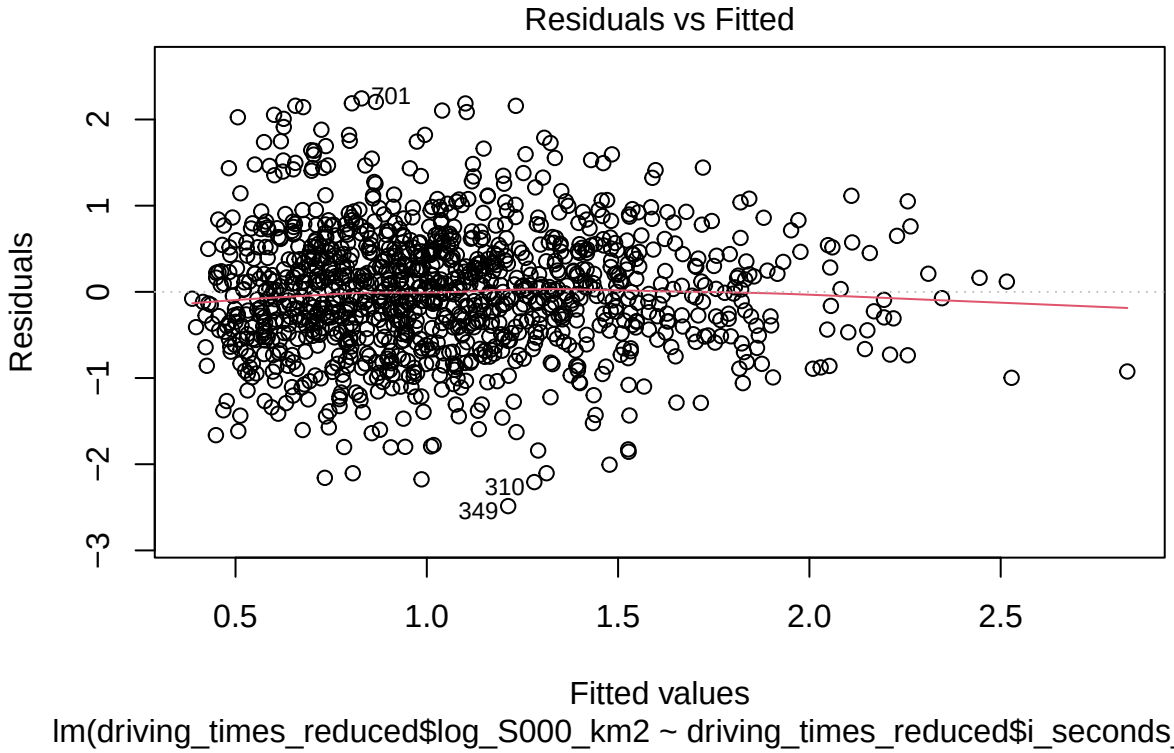
Driving

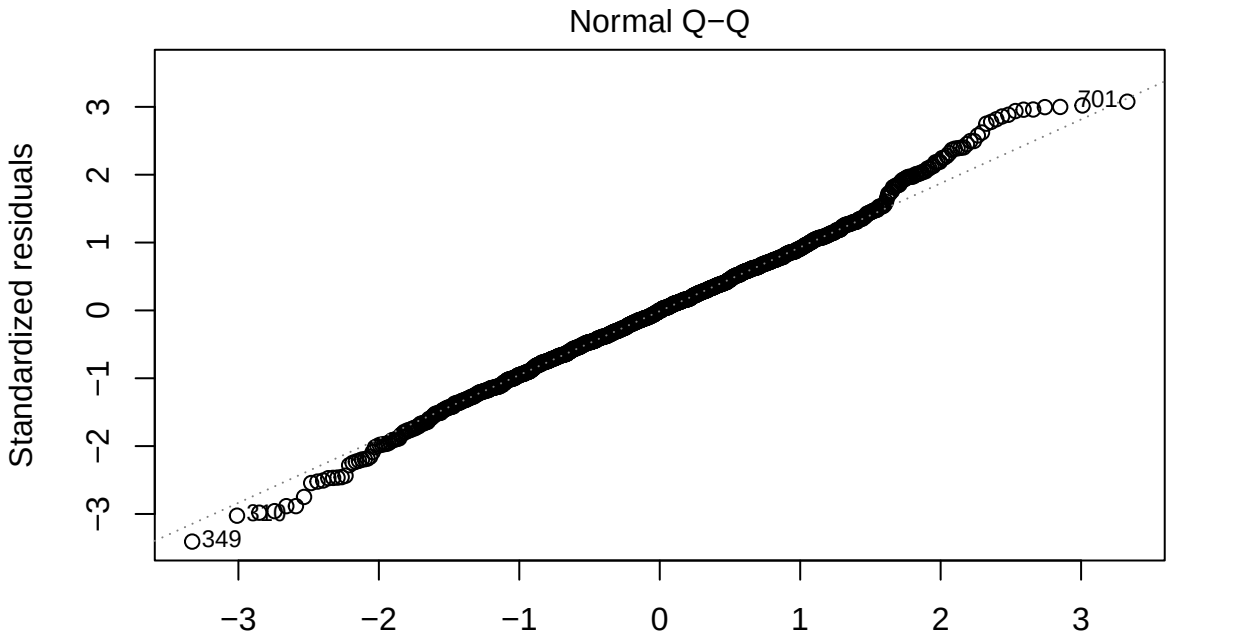
```
driving_model <- lm(driving_times_reduced$log_S000_km2 ~ driving_times_reduced$i_seconds_in_traffic)
summary(driving_model)
```

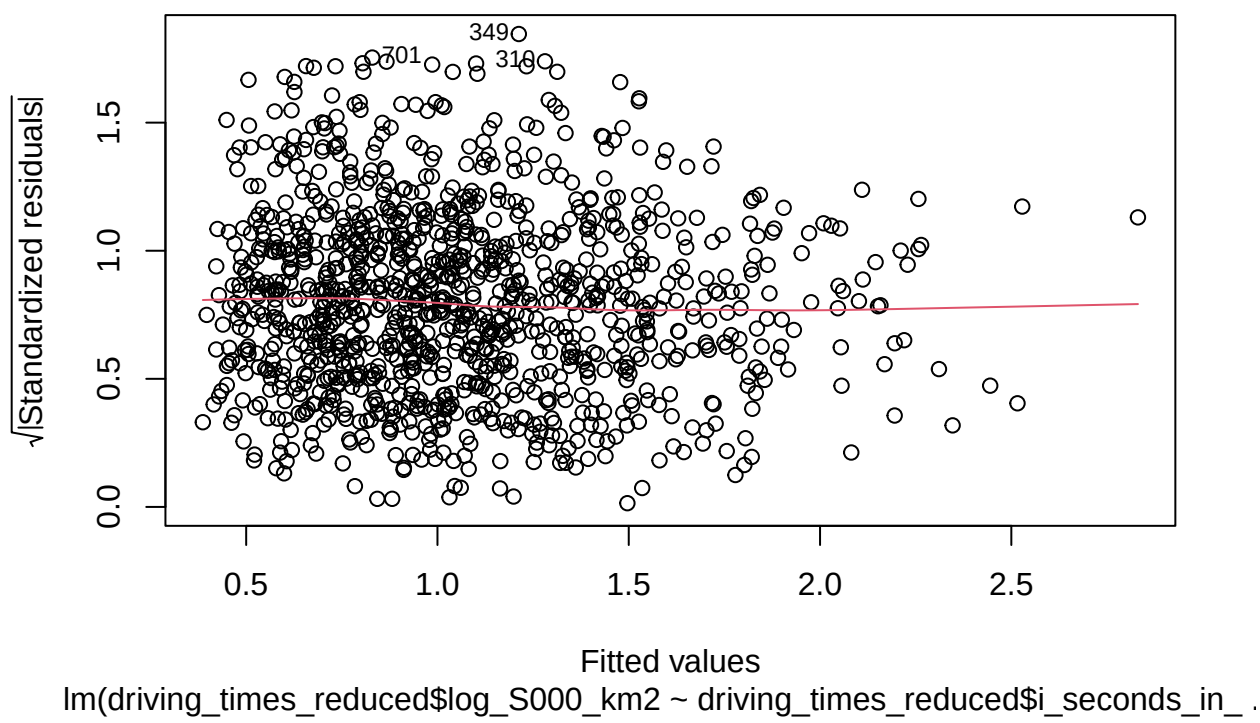
```
##
```

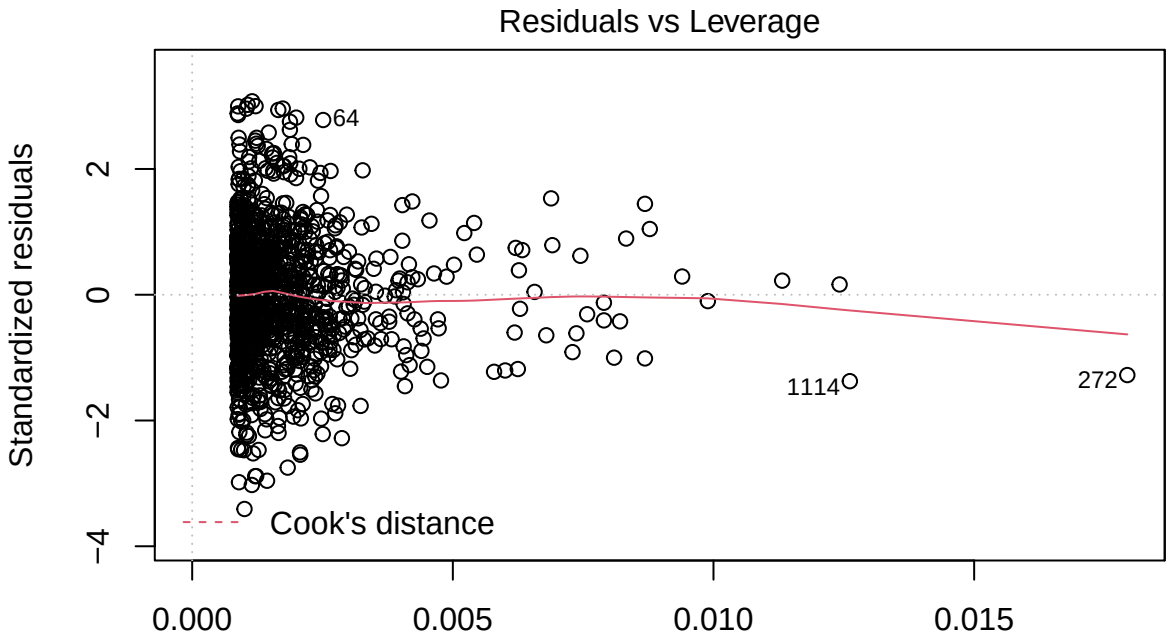
```
## Call:
## lm(formula = driving_times_reduced$log_S000_km2 ~ driving_times_reduced$i_seconds_in_traffic)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.48600 -0.47132 -0.00249  0.45555  2.24441
##
## Coefficients:
##                                           Estimate Std. Error t value
## (Intercept)                                -6.5733     0.4106  -16.01
## driving_times_reduced$i_seconds_in_traffic  19.0288     1.0226   18.61
##                                                    Pr(>|t|)
## (Intercept)                                <0.0000000000000002 ***
## driving_times_reduced$i_seconds_in_traffic <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.73 on 1146 degrees of freedom
## Multiple R-squared:  0.232,  Adjusted R-squared:  0.2314
## F-statistic: 346.2 on 1 and 1146 DF,  p-value: < 0.00000000000000022
```

```
plot(driving_model)
```




Normal Q−Q
Standardized residuals
Theoretical Quantiles
lm(driving_times_reduced$log_S000_km2 ~ driving_times_reduced$i_seconds_in_ .
701
349


Scale−Location
√|Standardized residuals|
Fitted values
lm(driving_times_reduced$log_S000_km2 ~ driving_times_reduced$i_seconds_in_ .
349
701
318

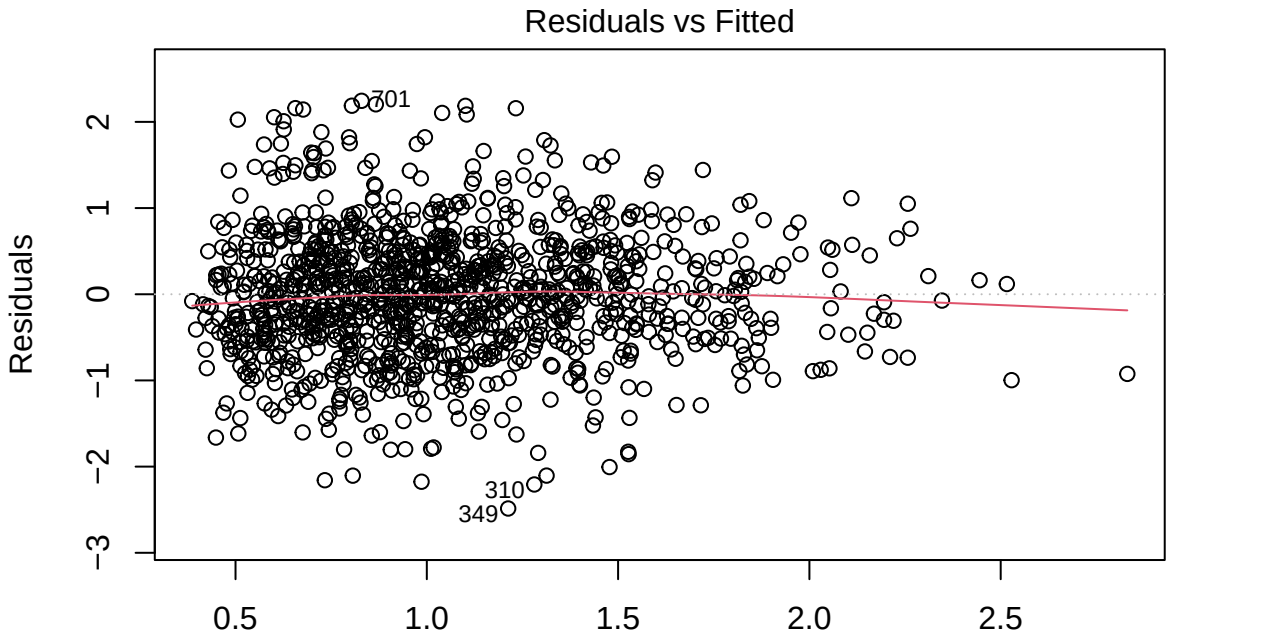


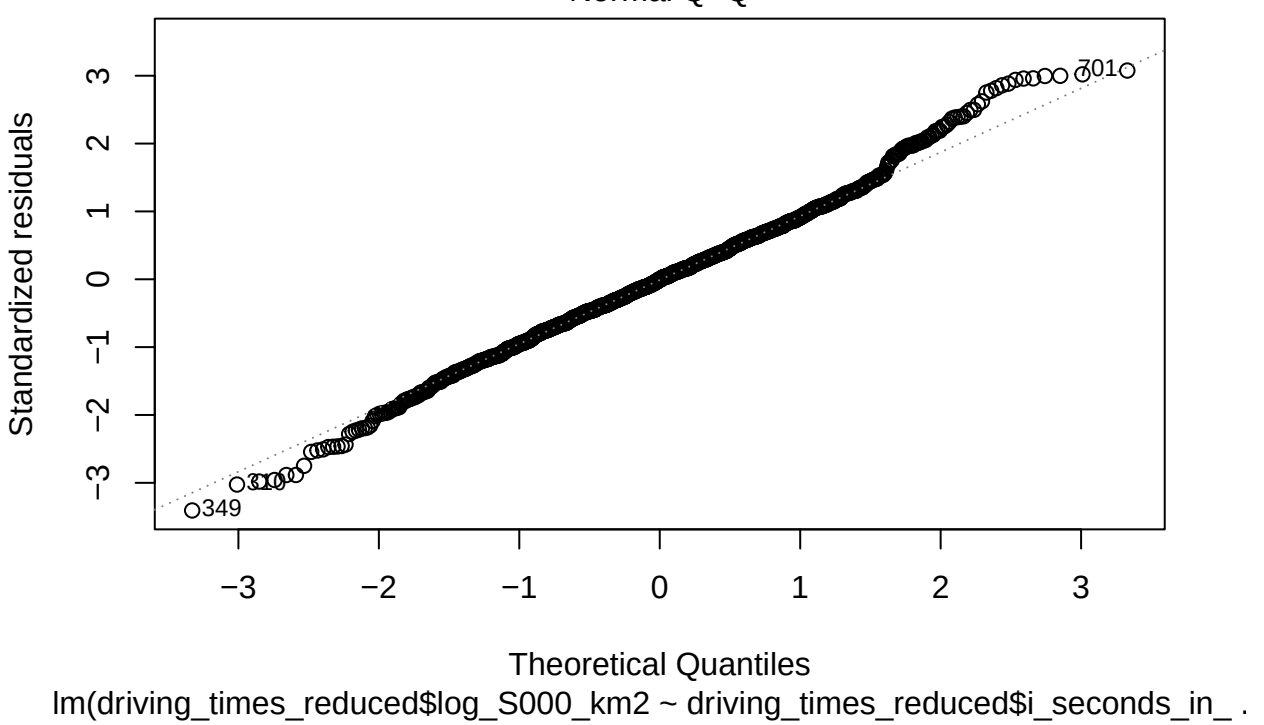
Walking

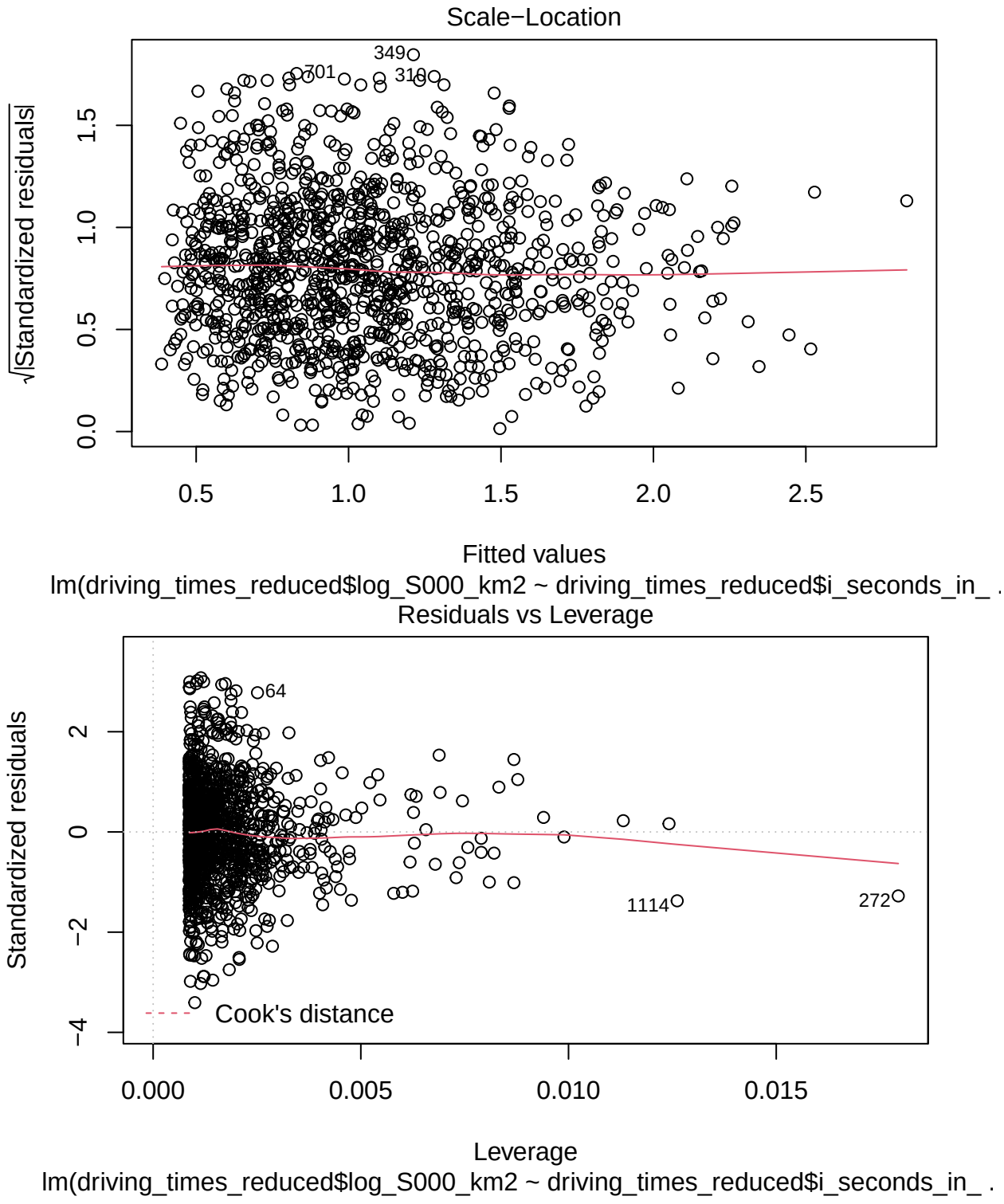
```
walking_model <- lm(walking_times_reduced$log_S000_km2 ~ walking_times_reduced$i_seconds_of_walking)
summary(walking_model)
```

```
##
## Call:
## lm(formula = walking_times_reduced$log_S000_km2 ~ walking_times_reduced$i_seconds_of_walking)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.09825 -0.41586 -0.02725  0.40889  2.14433
##
## Coefficients:
##                                             Estimate Std. Error t value
## (Intercept)                                   -7.014      0.312  -22.48
## walking_times_reduced$i_seconds_of_walking   23.692      0.914   25.92
##                                                      Pr(>|t|)
## (Intercept)                                  <0.0000000000000002 ***
## walking_times_reduced$i_seconds_of_walking <0.0000000000000002 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6614 on 1146 degrees of freedom
## Multiple R-squared:  0.3696, Adjusted R-squared:  0.3691
## F-statistic: 671.9 on 1 and 1146 DF,  p-value: < 0.00000000000000022
```

```
plot(driving_model)
```







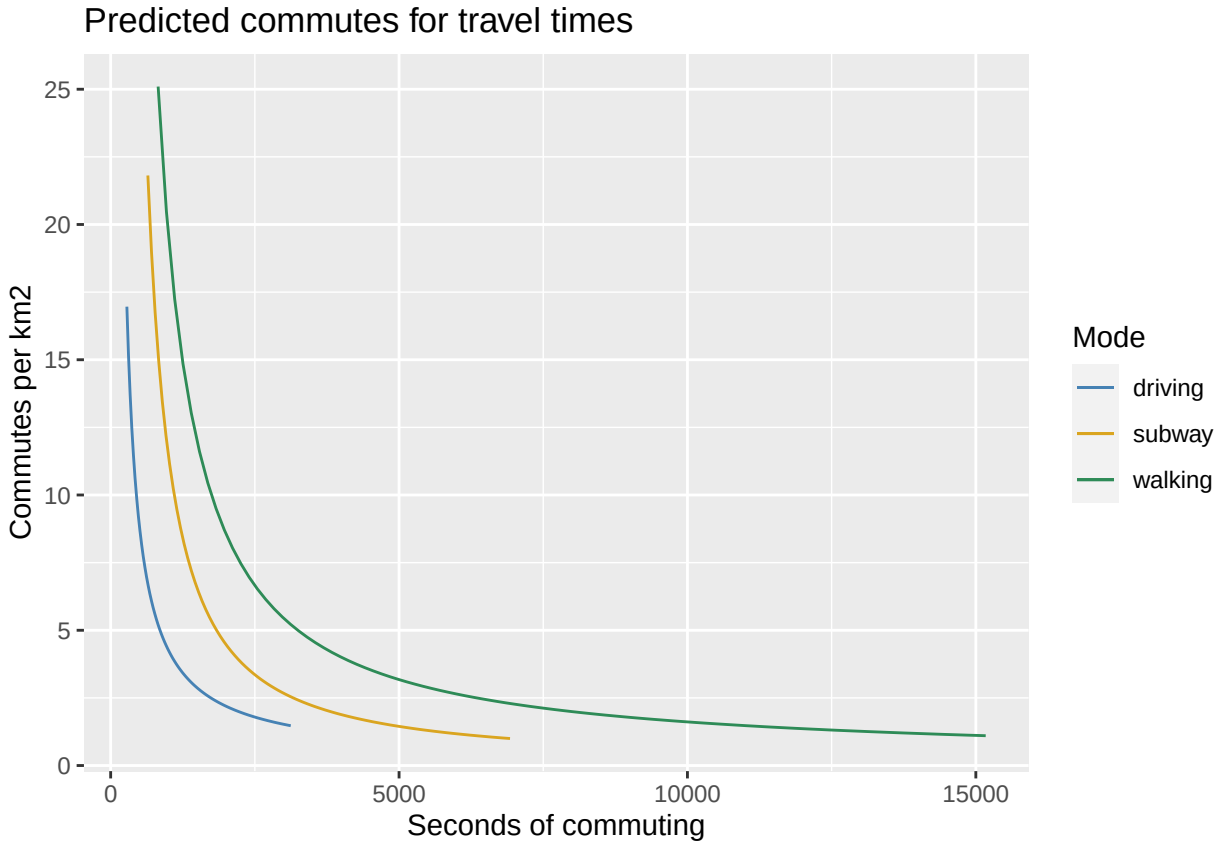
Multiple linear regression for all three factors

Equations plotted for all factors

```
subway_connected_eq <- function(t) exp(subway_connected_model$coefficients[[1]] + subway_connected_model
driving_eq <- function(t) exp(driving_model$coefficients[[1]] + driving_model$coefficients[[2]] / t^(1 /
```

```
walking_eq <- function(t) exp(walking_model$coefficients[[1]] + walking_model$coefficients[[2]] / t^(1 /
```

```
ggplot(
  dplyr::tibble(
    seconds = seq(from = 301, to = 15200, by = 14.9)
  ), aes(seconds)) +
  ggtitle("Predicted commutes for travel times") +
  xlab("Seconds of commuting") +
  ylab("Commutes per km2") +
  stat_function(fun = subway_connected_eq, aes(color = "subway"), xlim = c(min(subway_times_connected$se
  stat_function(fun = driving_eq, aes(color = "driving"), xlim = c(min(driving_times_reduced$seconds_in_
  stat_function(fun = walking_eq, aes(color = "walking"), xlim = c(min(walking_times_reduced$seconds_of_
  scale_color_manual("Mode", values=c("steelblue", "goldenrod", "seagreen"))
```




```
# The 12 minutes is the lowest round minute in each transportation mode's range
# 50 minutes is a reasonable high end of commute times, in each mode's range.

ggplot(
  dplyr::tibble(
    seconds = seq(from = 720, to = 3000, by = 14.9)
  ), aes(seconds)) +
  ggtitle(
    "Predicted commutes for travel times",
    subtitle = "Zoom to commutes between 12 and 50 minutes"
    ) +
  xlab("Seconds of commuting") +
  ylab("Commutes per km2") +
```

```
  stat_function(fun = subway_connected_eq, aes(color = "subway")) +
  stat_function(fun = driving_eq, aes(color = "driving")) +
  stat_function(fun = walking_eq, aes(color = "walking")) +
  scale_color_manual("Mode", values=c("steelblue", "goldenrod", "seagreen"))
```

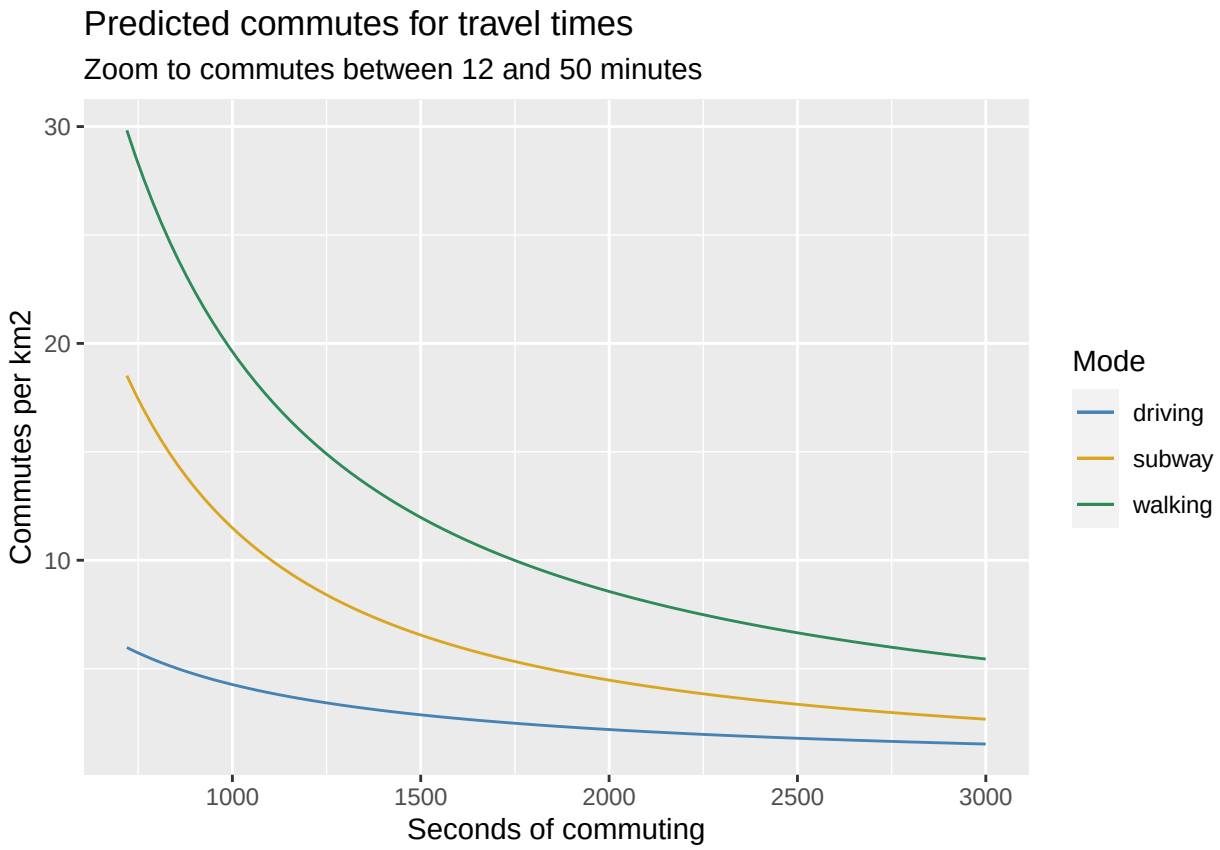


Table of values at 10, 25, 50

```
times_of_interest <- c(10, 25, 50)*60

## subway predictions
subway_predictions <- subway_connected_eq(times_of_interest)
subway_se <- summary(subway_connected_model)$sigma
subway_pred_low <- subway_predictions - (2*subway_se)
subway_pred_high <- subway_predictions + (2*subway_se)

## driving predictions
driving_prediction <- driving_eq(times_of_interest)
driving_se <- summary(driving_model)$sigma
driving_pred_low <- driving_prediction - (2*driving_se)
driving_pred_high <- driving_prediction + (2*driving_se)

## walking predictions
walking_predictions <- walking_eq(times_of_interest)
walking_se <- summary(walking_model)$sigma
walking_pred_low <- walking_predictions - (2*walking_se)
walking_pred_high <- walking_predictions + (2*walking_se)
```

```
predictions <- dplyr::tibble(
  Minutes = c("Ten", "Twenty Five", "Fifty"),
  subway_predictions,
  subway_pred_low,
  subway_pred_high,
  driving_prediction,
  driving_pred_low,
  driving_pred_high,
  walking_predictions,
  walking_pred_low,
  walking_pred_high
)

print(predictions)
```

```
## # A tibble: 3 x 10
##   Minutes     subway_p~1 subwa~2 subwa~3 drivi~4 drivi~5 drivi~6 walki~7 walki~8
##   <chr>            <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 Ten              24.3    23.0    25.7     7.24  5.78      8.70   37.9    36.6
## 2 Twenty Five       6.55    5.21    7.89    2.87  1.41      4.33   12.0    10.7
## 3 Fifty             2.67    1.33    4.01    1.52  0.0642    2.98    5.44    4.12
## # ... with 1 more variable: walking_pred_high <dbl>, and abbreviated variable
## #   names 1: subway_predictions, 2: subway_pred_low, 3: subway_pred_high,
## #   4: driving_prediction, 5: driving_pred_low, 6: driving_pred_high,
## #   7: walking_predictions, 8: walking_pred_low
```

Auto Correlation of Subway, Driving, and Walking

The data points removed for the linear regression are returned for the correlation. For driving and walking, they are returned without further modification. For the subway analysis, any trips that were made without using the subway are giving a value of 0. This reflects that the two NTAs involved in the trip are not actually neighbors through the subway definition.

In addition to the three infrastructure neighborhood definitions, a Queens contiguity analysis is run for reference.

The natural log of job counts per km2 is used as the value for each NTA.

Global Moran's I

```
subway_times <- subway_times %>%
  dplyr::mutate(
    i_c_seconds_in_transit = ifelse(subway_lines$line_count > 0, i_seconds_in_transit, 0),
  )

subway_graph <- subway_times %>%
  dplyr::select(
    c(
      nta_one,
      nta_two,
      i_c_seconds_in_transit
    )
  ) %>%
  dplyr::rename(
    from = nta_one,
```

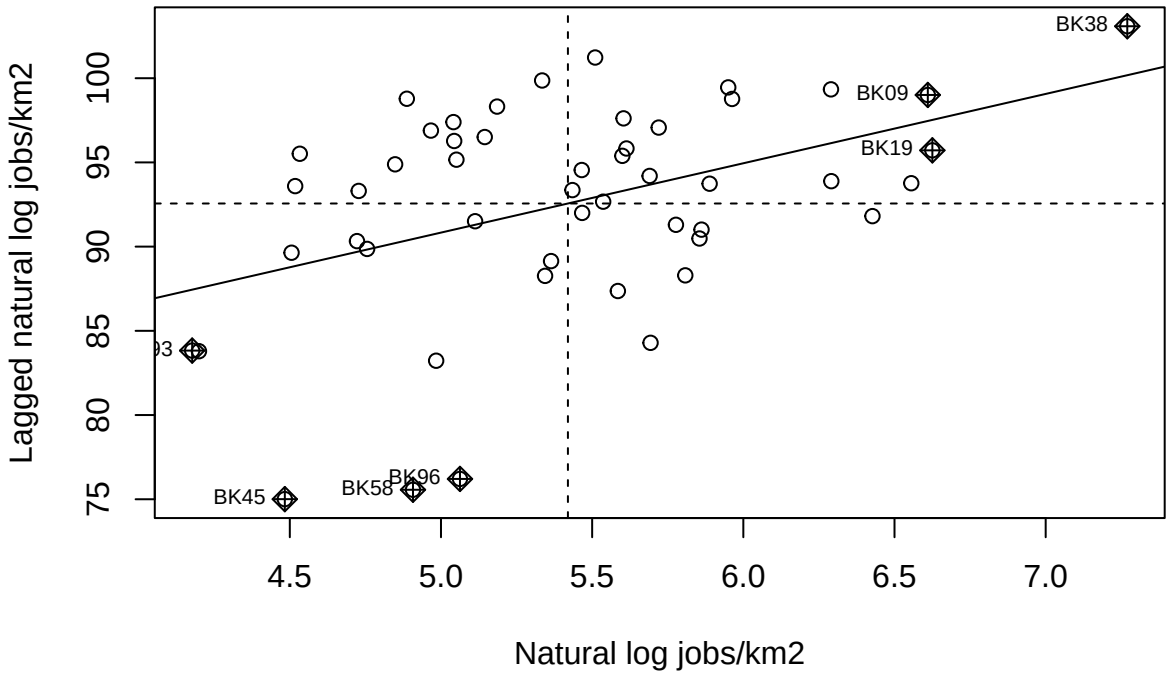
```
    to = nta_two,
    weight = i_c_seconds_in_transit,
  ) %>%
  igraph::graph.data.frame(
    directed = FALSE
  )

subway_weights <- subway_graph %>%
  igraph::as_adjacency_matrix(attr = "weight") %>%
  spdep::mat2listw()

driving_weights <- driving_times %>%
  dplyr::select(
    c(
      nta_one,
      nta_two,
      i_seconds_in_traffic
    )
  ) %>%
  dplyr::rename(
    from = nta_one,
    to = nta_two,
    weight = i_seconds_in_traffic
  ) %>%
  igraph::graph.data.frame(
    directed = FALSE
  ) %>%
  igraph::as_adjacency_matrix(attr = "weight") %>%
  spdep::mat2listw()

walking_weights <- walking_times %>%
  dplyr::select(
    c(
      nta_one,
      nta_two,
      i_seconds_of_walking
    )
  ) %>%
  dplyr::rename(
    from = nta_one,
    to = nta_two,
    weight = i_seconds_of_walking
  ) %>%
  igraph::graph.data.frame(
    directed = FALSE
  ) %>%
  igraph::as_adjacency_matrix(attr = "weight") %>%
  spdep::mat2listw()

queen_weights <- job_counts %>%
  spdep::poly2nb(c("w_nta_code")) %>%
  spdep::nb2listw(zero.policy = TRUE)
```

```
subway_global_morans <- spdep::moran.test(
  job_counts$log_S000_km2,
  subway_weights,
  zero.policy = TRUE,
)
print(subway_global_morans)
```

Subway

```
##
##  Moran I test under randomisation
##
## data:  job_counts$log_S000_km2
## weights: subway_weights
##
## Moran I statistic standard deviate = -3.5321, p-value = 0.9998
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##      -0.044946546      -0.020408163       0.000048265
```

```
spdep::moran.plot(
  job_counts$log_S000_km2,
  subway_weights,
  zero.policy = TRUE,
  xlab = "Natural log jobs/km2",
  ylab = "Lagged natural log jobs/km2"
)
```



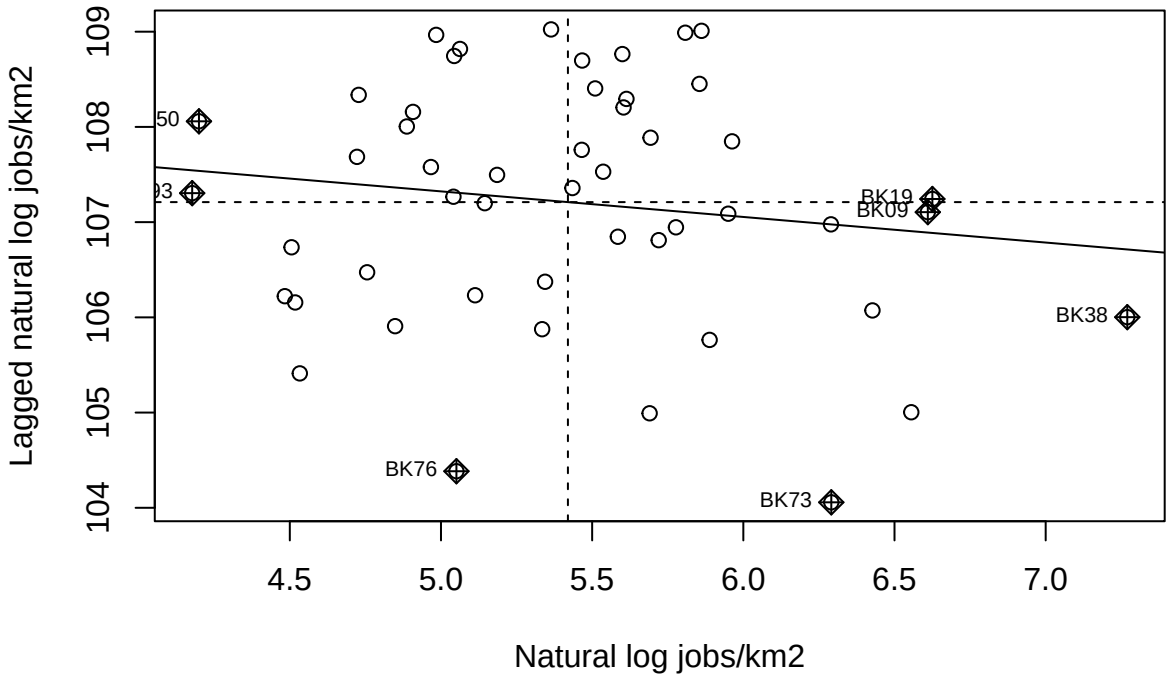
Driving

```
driving_global_morans <- spdep::moran.test(
  job_counts$log_S000_km2,
```

```
  driving_weights,
  zero.policy = TRUE,
)
print(driving_global_morans)
```

```
##
##  Moran I test under randomisation
##
## data:  job_counts$log_S000_km2
## weights: driving_weights
##
## Moran I statistic standard deviate = 6.6184, p-value = 0.00000000001815
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##   -0.009846545704   -0.020408163265    0.000002546533
```

```
spdep::moran.plot(
  job_counts$log_S000_km2,
  driving_weights,
  zero.policy = TRUE,
  xlab = "Natural log jobs/km2",
  ylab = "Lagged natural log jobs/km2"
)
```

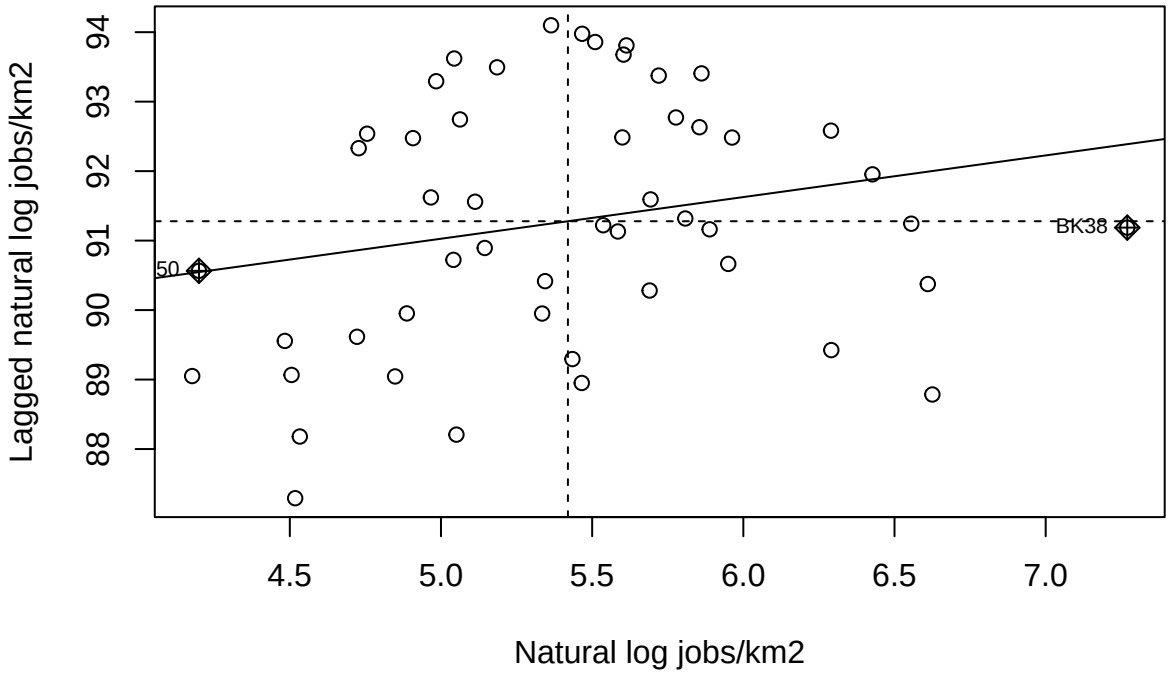



```
walking_global_morans <- spdep::moran.test(
  job_counts$log_S000_km2,
  walking_weights,
  zero.policy = TRUE,
)
print(walking_global_morans)
```

Walking

```
##
##  Moran I test under randomisation
##
## data:  job_counts$log_S000_km2
## weights: walking_weights
##
## Moran I statistic standard deviate = 6.2078, p-value = 0.0000000002687
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##   -0.008981502858   -0.020408163265    0.000003388162
```

```
spdep::moran.plot(
  job_counts$log_S000_km2,
  walking_weights,
  zero.policy = TRUE,
  xlab = "Natural log jobs/km2",
  ylab = "Lagged natural log jobs/km2"
)
```



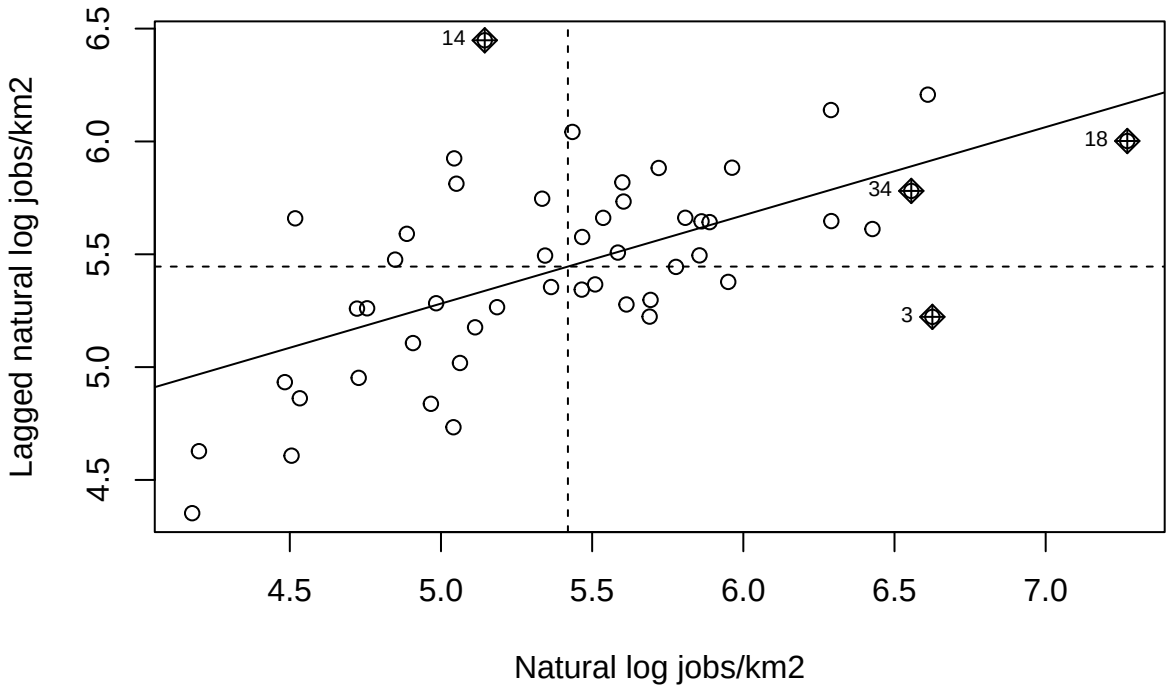
Queens

```
queen_global_morans <- spdep::moran.test(
  job_counts$log_S000_km2,
  queen_weights,
  zero.policy = TRUE,
)
print(queen_global_morans)
```

```
##
##  Moran I test under randomisation
##
## data:  job_counts$log_S000_km2
```

```
## weights: queen_weights
##
## Moran I statistic standard deviate = 4.4715, p-value = 0.000003884
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##       0.391099443      -0.020408163       0.008469485
```

```
spdep::moran.plot(
  job_counts$log_S000_km2,
  queen_weights,
  zero.policy = TRUE,
  xlab = "Natural log jobs/km2",
  ylab = "Lagged natural log jobs/km2"
)
```



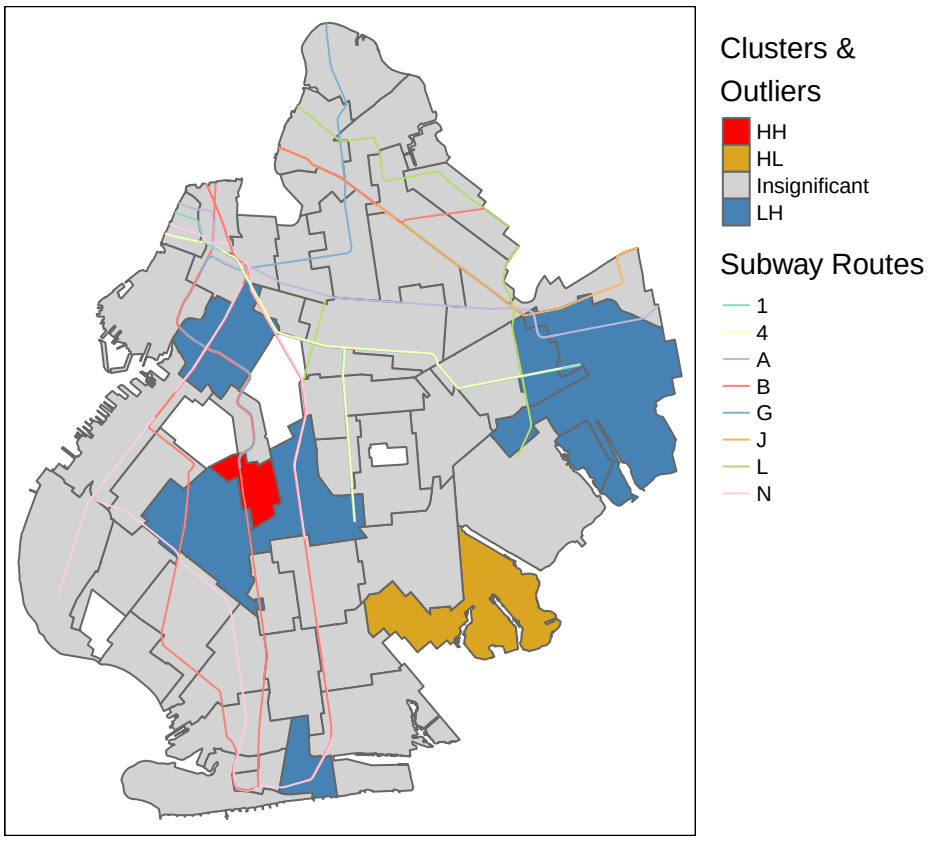
LISA

```
avg_jobs <- mean(job_counts$log_S000_km2)

classify_co_types <- function(mode_lisa, l_job_counts, avg_job_count) {
  mode_lisa %>%
    tibble::as_tibble() %>%
    magrittr::set_colnames(
      c("Ii","E.Ii","Var.Ii","Z.Ii","Pr(z > 0)")
    ) %>%
    dplyr::mutate(
      co_type = dplyr::case_when(
        `Pr(z > 0)` <= 0.05 &
          Ii >= 0 &
          l_job_counts >= avg_job_count ~ "HH",
        `Pr(z > 0)` <= 0.05 &
```

```
          Ii >= 0 &
          l_job_counts <  avg_job_count ~ "LL",
        `Pr(z > 0)` <= 0.05 &
          Ii < 0 &
          l_job_counts >= avg_job_count ~ "HL",
        `Pr(z > 0)` <= 0.05 &
          Ii < 0 &
          l_job_counts < avg_job_count ~ "LH"
      )
    )
}

subway_lisa <- spdep::localmoran(
  job_counts$log_S000_km2,
  subway_weights,
  zero.policy = TRUE,
  na.action = na.omit
)

driving_lisa <- spdep::localmoran(
  job_counts$log_S000_km2,
  driving_weights,
  zero.policy = TRUE,
  na.action = na.omit
)

walking_lisa <- spdep::localmoran(
  job_counts$log_S000_km2,
  walking_weights,
  zero.policy = TRUE,
  na.action = na.omit
)

queen_lisa <- spdep::localmoran(
  job_counts$log_S000_km2,
  queen_weights,
  zero.policy = TRUE,
  na.action = na.omit
)

subway_classes <- classify_co_types(subway_lisa, job_counts$log_S000_km2, avg_jobs)
driving_classes <- classify_co_types(driving_lisa, job_counts$log_S000_km2, avg_jobs)
walking_classes <- classify_co_types(walking_lisa, job_counts$log_S000_km2, avg_jobs)
queen_classes <- classify_co_types(queen_lisa, job_counts$log_S000_km2, avg_jobs)

subway_bk_nta_border <- bk_nta_border %>%
  dplyr::mutate(
    co_type = ifelse(is.na(subway_classes$co_type), "Insignificant", subway_classes$co_type)
  )

driving_bk_nta_border <- bk_nta_border %>%
  dplyr::mutate(
    co_type = ifelse(is.na(driving_classes$co_type), "Insignificant",  driving_classes$co_type)
```

```
  )

walking_bk_nta_border <- bk_nta_border %>%
  dplyr::mutate(
    co_type = ifelse(is.na(walking_classes$co_type), "Insignificant", walking_classes$co_type)
  )

queen_bk_nta_border <- bk_nta_border %>%
  dplyr::mutate(
    co_type = ifelse(is.na(queen_classes$co_type), "Insignificant", queen_classes$co_type)
  )
```

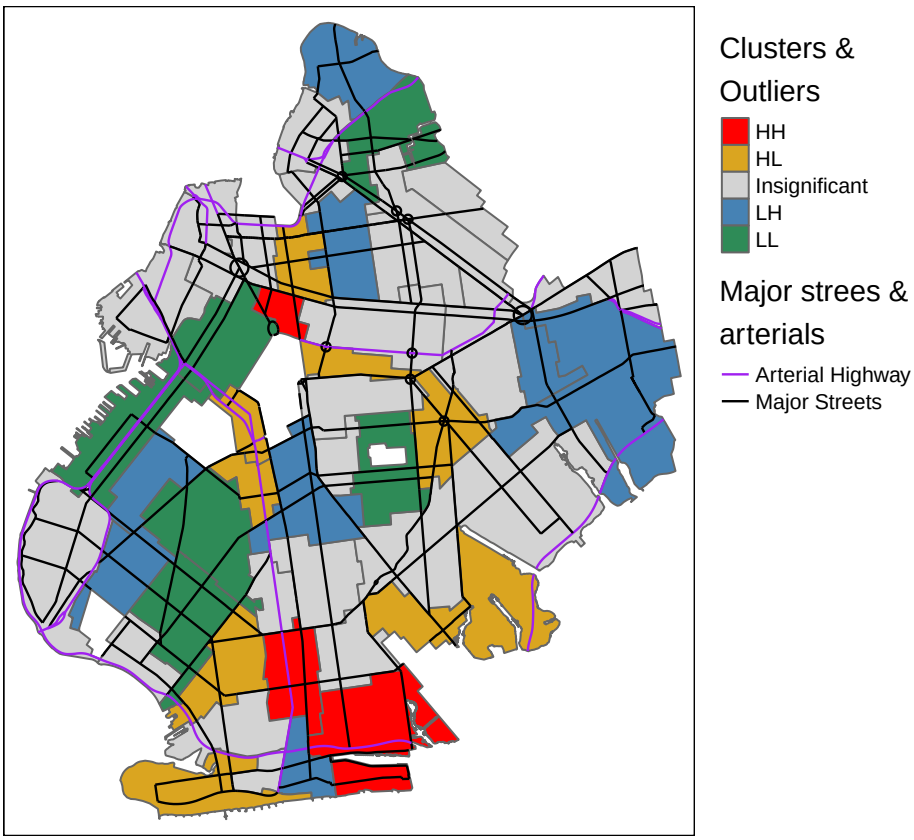
```
tmap::tm_shape(subway_bk_nta_border) +
  tm_polygons(
    col = "co_type",
    palette = c("red", "goldenrod", "lightgrey", "steelblue"),
    title = "Clusters &\nOutliers"
  ) +
  tmap::tm_shape(bk_subways) +
  tmap::tm_lines(
    col = "rt_symbol",
    title.col = "Subway Routes"
  ) +
  tmap::tm_layout(
    legend.outside = TRUE
  )
```



Subway

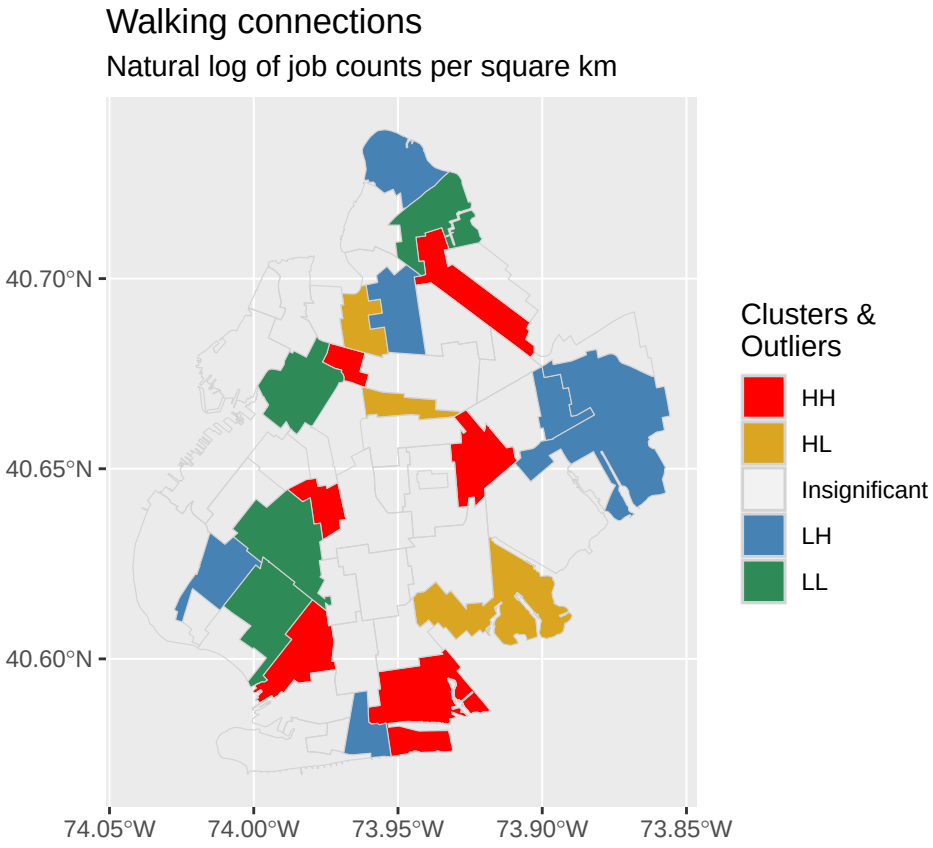
Driving

```
tmap::tm_shape(driving_bk_nta_border) +
  tm_polygons(
    col = "co_type",
    palette = c("red", "goldenrod", "lightgrey", "steelblue", "seagreen"),
    title = "Clusters &\nOutliers"
  ) +
  tmap::tm_shape(bk_major_roads) +
  tmap::tm_lines(
    col = "route_type",
    lwd = 1.2,
    palette = c("purple", "black"),
    title.col = "Major strees &\narterials"
  ) +
  tmap::tm_layout(
    legend.outside = TRUE
  )
```



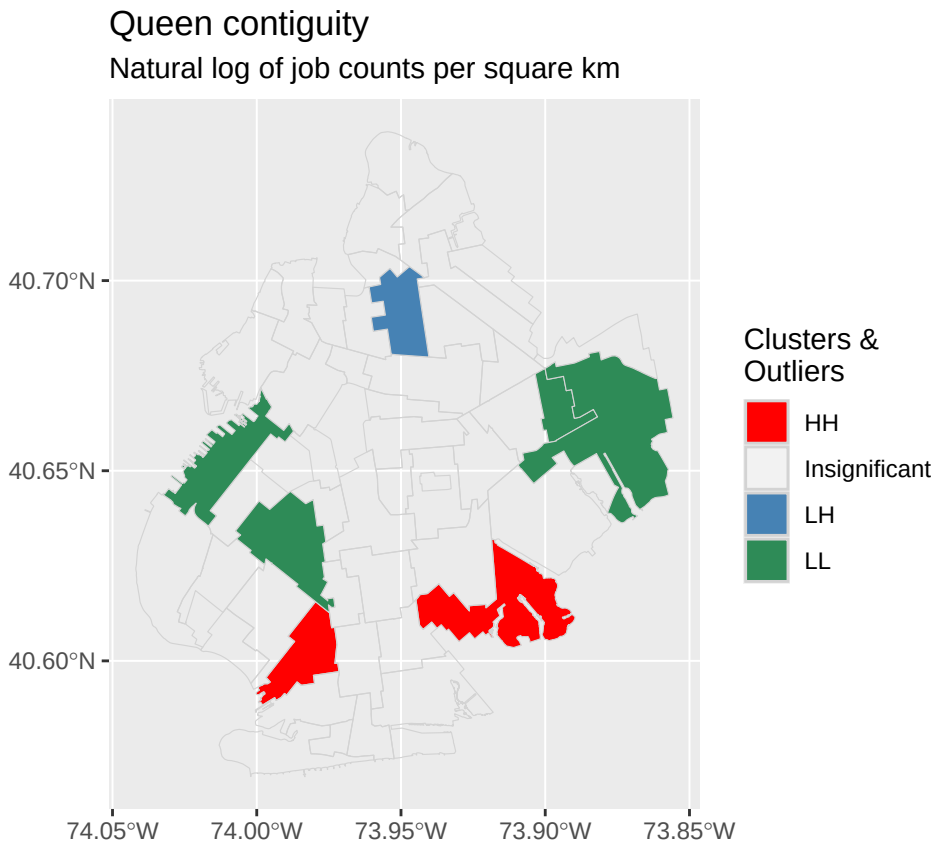
Walking

```
ggplot(walking_bk_nta_border) +
  geom_sf(aes(fill = co_type), col = 'lightgrey') +
  scale_fill_manual(
    values = c("red", "goldenrod", "NA", "steelblue", "seagreen"),
    name = "Clusters & \nOutliers"
  ) +
  labs(
    title = "Walking connections",
    subtitle = "Natural log of job counts per square km"
  )
```



Queen Contiguity

```
ggplot(queen_bk_nta_border) +
  geom_sf(aes(fill = co_type), col = 'lightgrey') +
  scale_fill_manual(
    values = c("red", "NA", "steelblue", "seagreen"),
    name = "Clusters & \nOutliers"
  ) +
  labs(
    title = "Queen contiguity",
    subtitle = "Natural log of job counts per square km"
  )
```



Network autocorrelation

```
## Utility function that builds the edges between 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_nta_one <- stringr::str_sub(from_node, 1, 4)
    from_nta_two <- stringr::str_sub(from_node, 5, 8)
    for(j in offset:nodes_count){
      to_node <- nodes[j]
      are_neighbors <- stringr::str_detect(to_node, from_nta_one) | stringr::str_detect(to_node, from_n
      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))
}
```

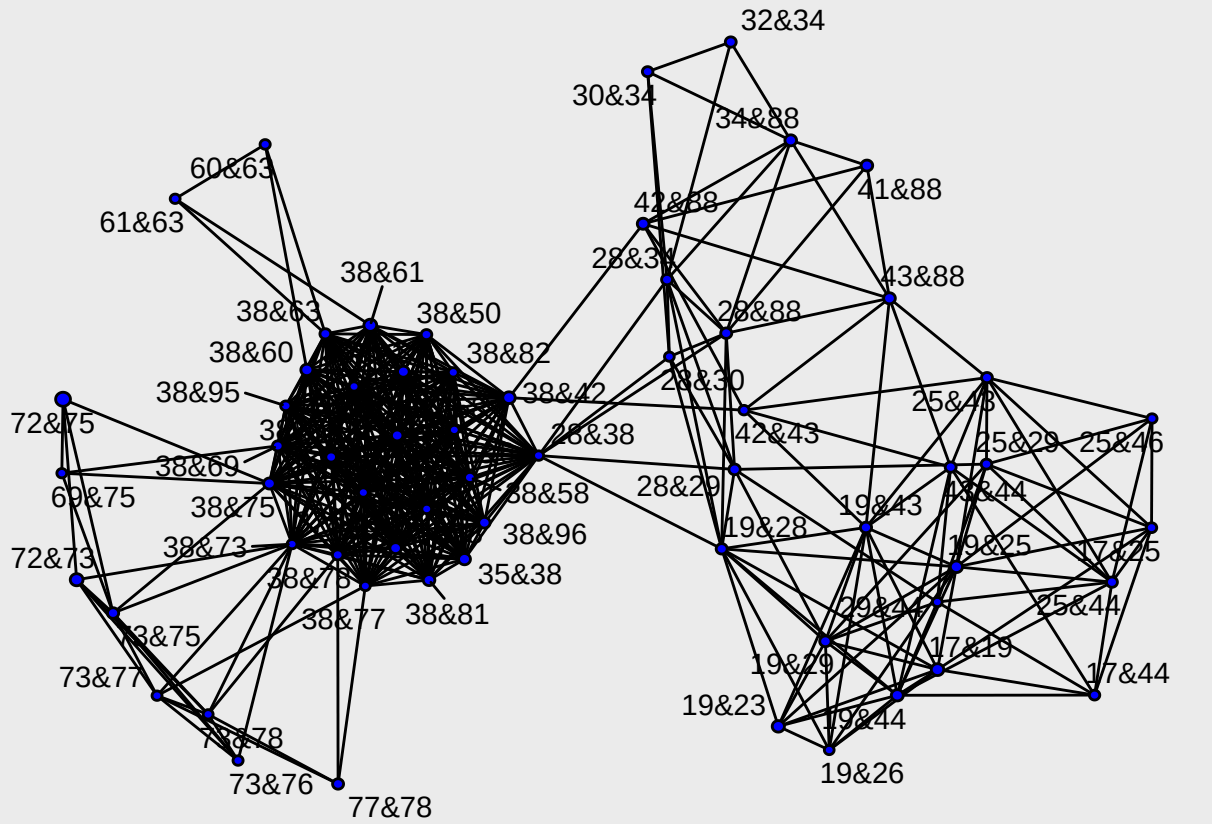
```
commute_nodes <- commute_counts %>%
  dplyr::select(
    c(trip, log_S000_km2)
```

```
  ) %>%
  dplyr::rename(
    name = trip
  )
commute_edges <- build_edges(commute_nodes$name)
commute_network <- tidygraph::tbl_graph(
  nodes = commute_nodes,
  edges = commute_edges,
  directed = FALSE
)
```

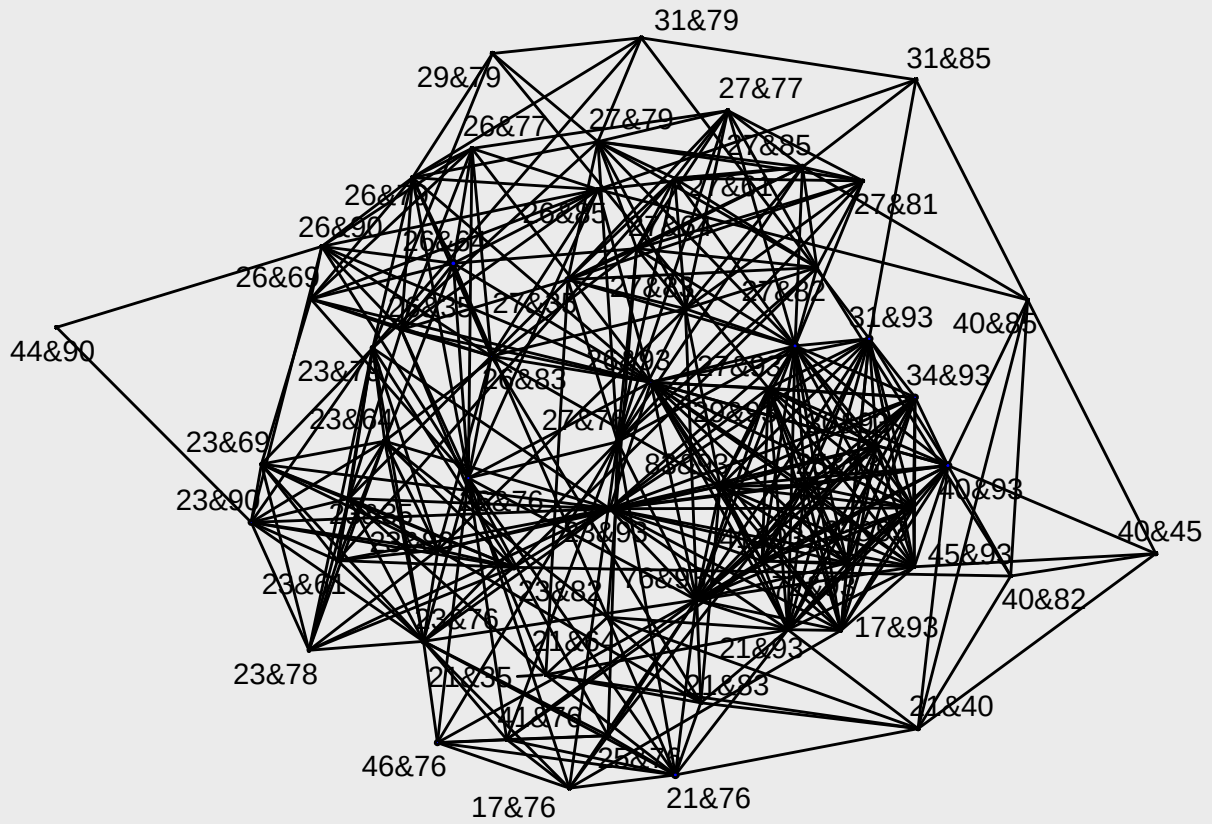
```
commute_nodes_top <- commute_nodes %>%
  dplyr::slice_max(
    order_by = log_S000_km2,
    n = 62
  )

commute_edges_top <- build_edges(commute_nodes_top$name)
commute_network_top <- tidygraph::tbl_graph(
  nodes = commute_nodes_top,
  edges = commute_edges_top,
  directed = FALSE
)

total_trips_top = sum(commute_nodes_top$log_S000_km2)
ggraph::ggraph(
  commute_network_top,
  layout = "stress"
) +
  geom_edge_link() +
  geom_node_circle(aes(r = commute_nodes_top$log_S000_km2 / 100), fill = "blue") +
  geom_node_text(aes(label = stringr::str_c(stringr::str_sub(name, 3,4), '&', stringr::str_sub(name, 7,8
```

Visualization of network's complement and most popular trips

```
## 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.
```




```
commute_nodes_bottom <- commute_nodes %>%
  dplyr::slice_min(
    order_by = log_S000_km2,
    n = 62
  )

commute_edges_bottom <- build_edges(commute_nodes_bottom$name)
commute_network_bottom <- tidygraph::tbl_graph(
  nodes = commute_nodes_bottom,
  edges = commute_edges_bottom,
  directed = FALSE
)

total_trips_bottom = sum(commute_nodes_bottom$log_S000_km2)
ggraph::ggraph(
  commute_network_bottom,
  layout = "stress"
) +
  geom_edge_link() +
  geom_node_circle(aes(r = commute_nodes_bottom$log_S000_km2 / 100), fill = "blue") +
  geom_node_text(aes(label = stringr::str_c(stringr::str_sub(name, 3,4), '&', stringr::str_sub(name, 7,8
```

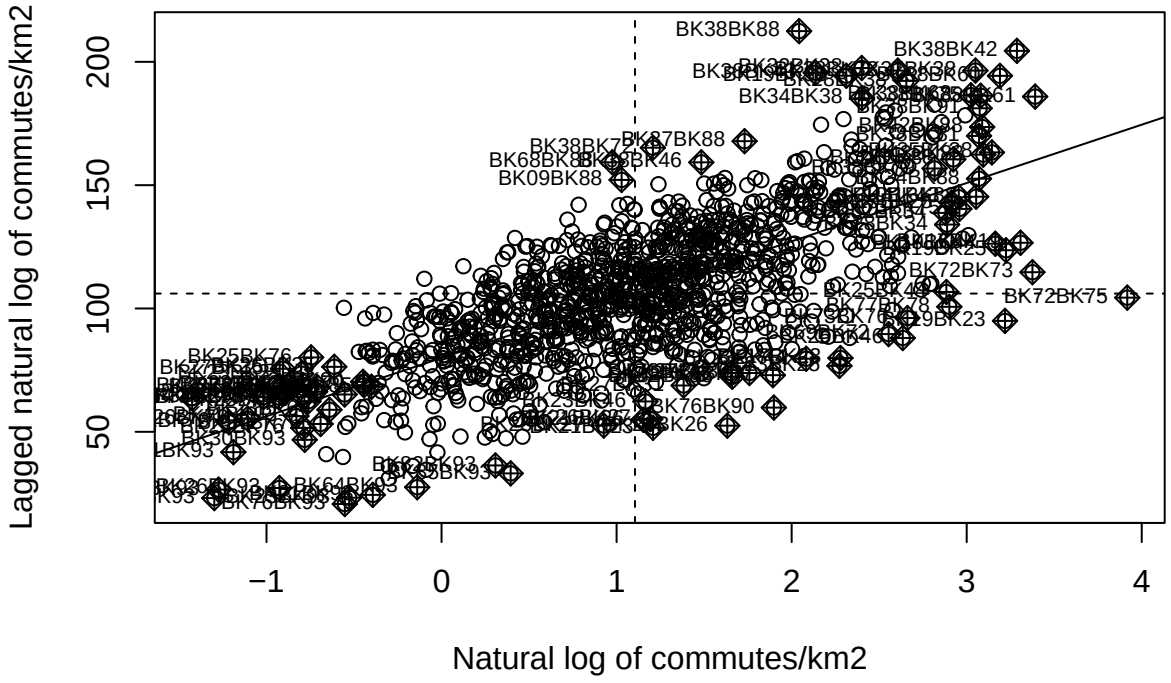



```
commute_weights <- commute_network %>%
  igraph::as_adj() %>%
  spdep::mat2listw()

commute_global_morans <- spdep::moran.test(
  commute_nodes$log_S000_km2,
  commute_weights,
  zero.policy = TRUE
)

print(commute_global_morans)
```

Global Moran's I

```
##
##  Moran I test under randomisation
##
## data:  commute_nodes$log_S000_km2
## weights: commute_weights
##
## Moran I statistic standard deviate = 62.413, p-value <
## 0.00000000000000022
## alternative hypothesis: greater
## sample estimates:
## Moran I statistic       Expectation          Variance
##     0.24627775555    -0.00081699346     0.00001567388
```

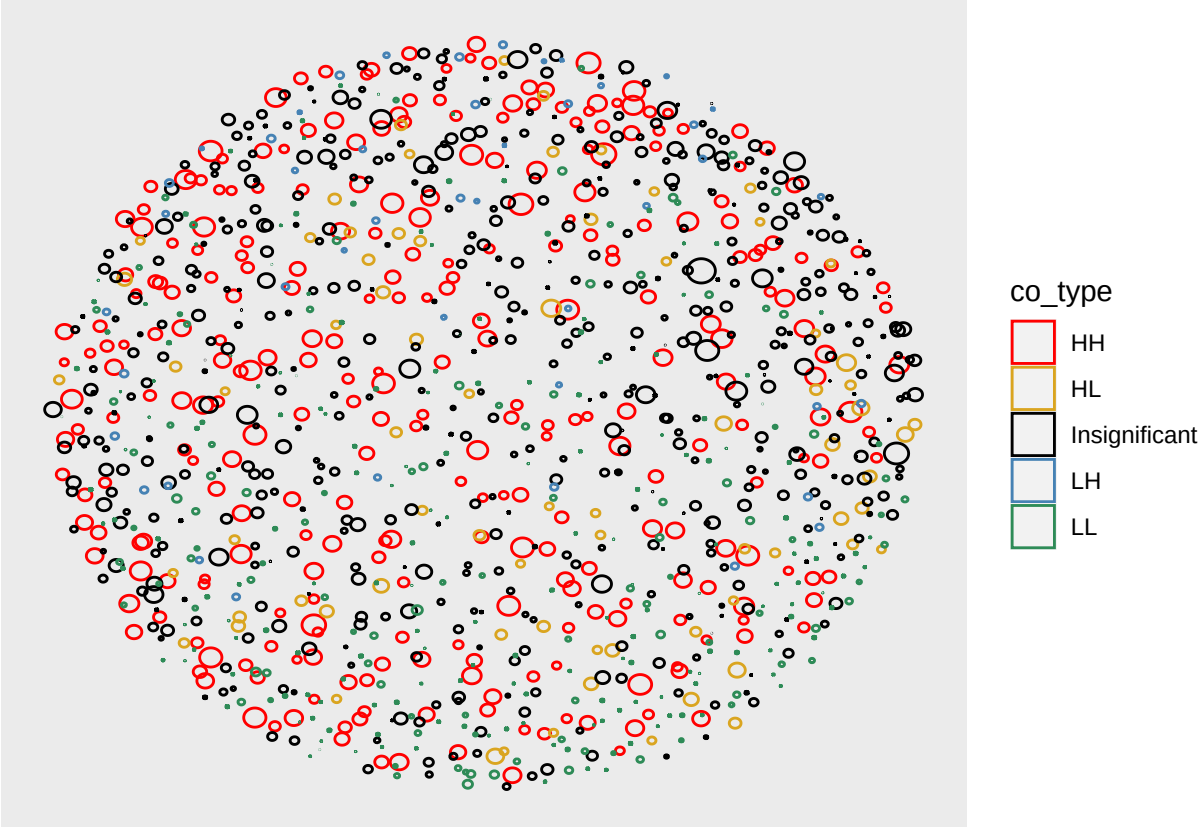
```
spdep::moran.plot(
  commute_nodes$log_S000_km2,
  commute_weights,
  zero.policy = TRUE,
  xlab = "Natural log of commutes/km2",
  ylab = "Lagged natural log of commutes/km2"
)
```




```
commute_lisa <- spdep::localmoran(
  commute_nodes$log_S000_km2,
  commute_weights,
  zero.policy = TRUE,
  na.action = na.omit
)

avg_commute_count <- mean(commute_nodes$log_S000_km2)

commute_classes <- classify_co_types(commute_lisa, commute_nodes$log_S000_km2, avg_commute_count)

commute_network_cluster <- commute_network %>%
  tidygraph::activate(nodes) %>%
  dplyr::mutate(
    co_type  = commute_classes$co_type %>% tidyr::replace_na("Insignificant")
  )

commute_network_cluster_sig <- commute_network_cluster %>%
  dplyr::filter(co_type != "Insignificant")
```

```
ggraph::ggraph(
  commute_network_cluster,
```

```
  layout = "stress"
) +
  ggraph::geom_node_circle(
    aes(r = log_S000_km2 / 75, color = co_type)
    ) +
  scale_color_manual(values = c("red", "goldenrod", "black", "steelblue", "seagreen"))
```



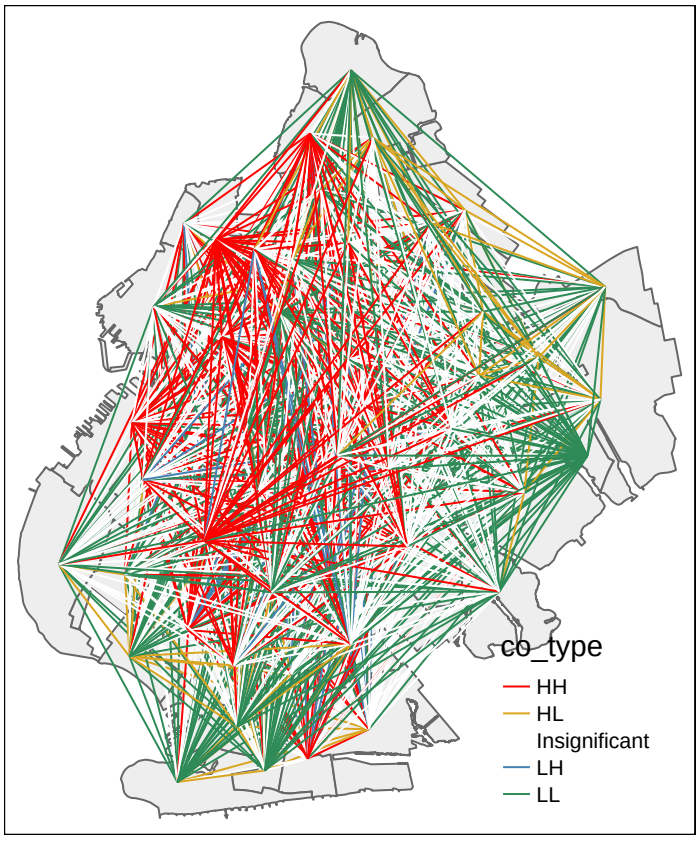
LISA

```
commute_count_lines <- commute_counts %>%
  dplyr::mutate(
    center_one = sf::st_point_on_surface(geometry_one),
    center_two = sf::st_point_on_surface(geometry_two)
  ) %>%
  dplyr::mutate(
    geometry = sf::st_cast(sf::st_union(center_one, center_two), "LINESTRING")
  ) %>%
  dplyr::select(
    c(trip, log_S000_km2, geometry)
  ) %>%
  sf::st_as_sf() %>%
  dplyr::mutate(
    co_type  = commute_classes$co_type %>% tidyr::replace_na("Insignificant")
  )
```

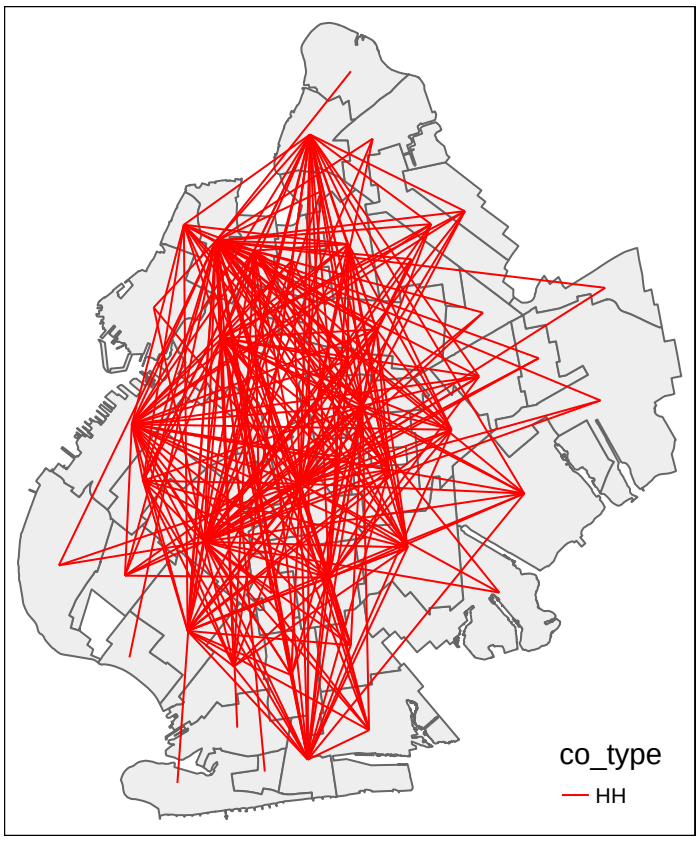
```
## Warning in st_point_on_surface.sfc(geometry_one): st_point_on_surface may not
## give correct results for longitude/latitude data
```

```
## Warning in st_point_on_surface.sfc(geometry_two): st_point_on_surface may not
## give correct results for longitude/latitude data
```

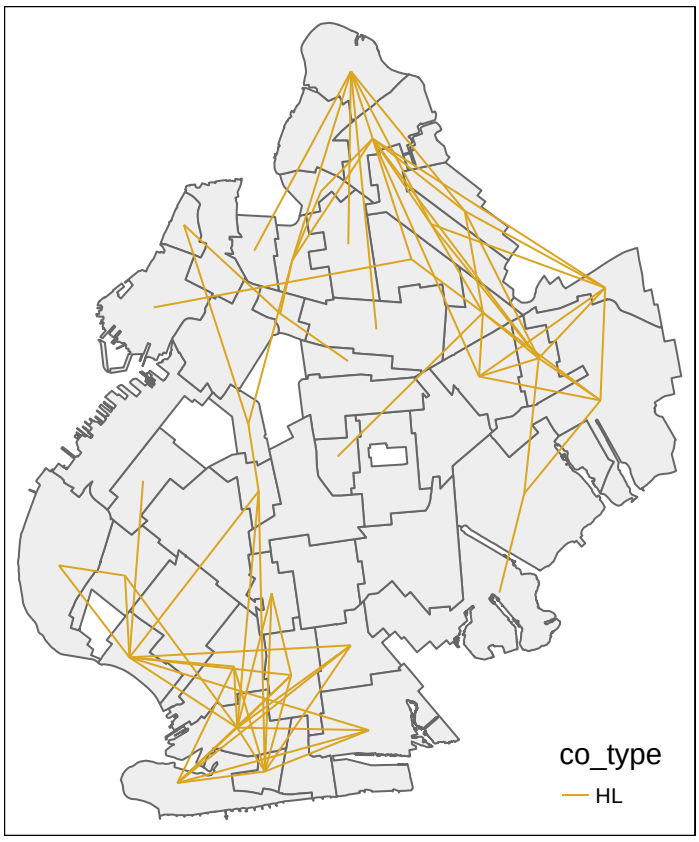
```
tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("red", "goldenrod", "white", "steelblue", "seagreen")
  )
```



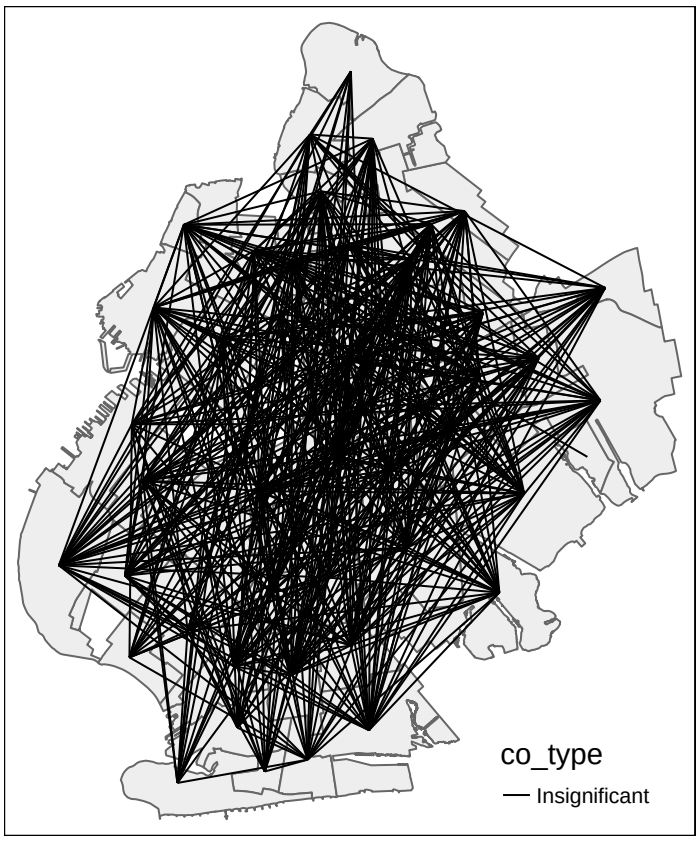

```
tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines %>% filter(co_type == "HH")) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("red")
  )
```



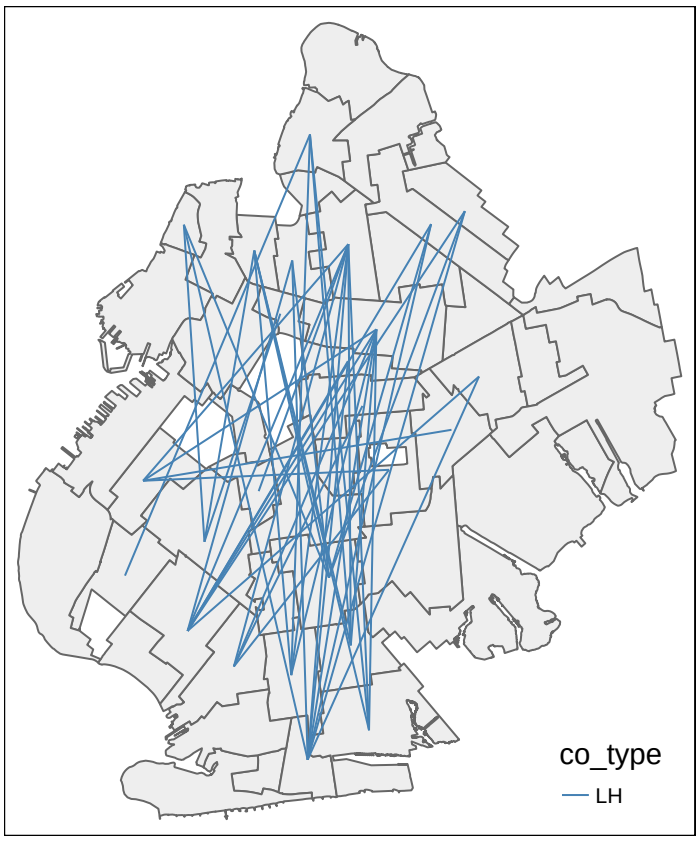

```
tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines %>% filter(co_type == "HL")) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("goldenrod")
  )
```



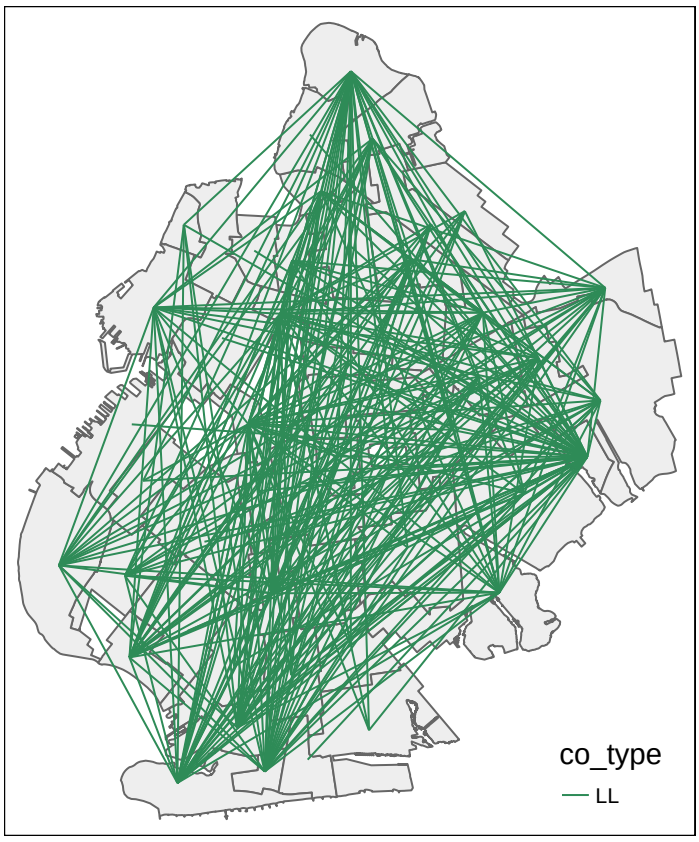

```
tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines %>% filter(co_type == "Insignificant")) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("black")
  )
```



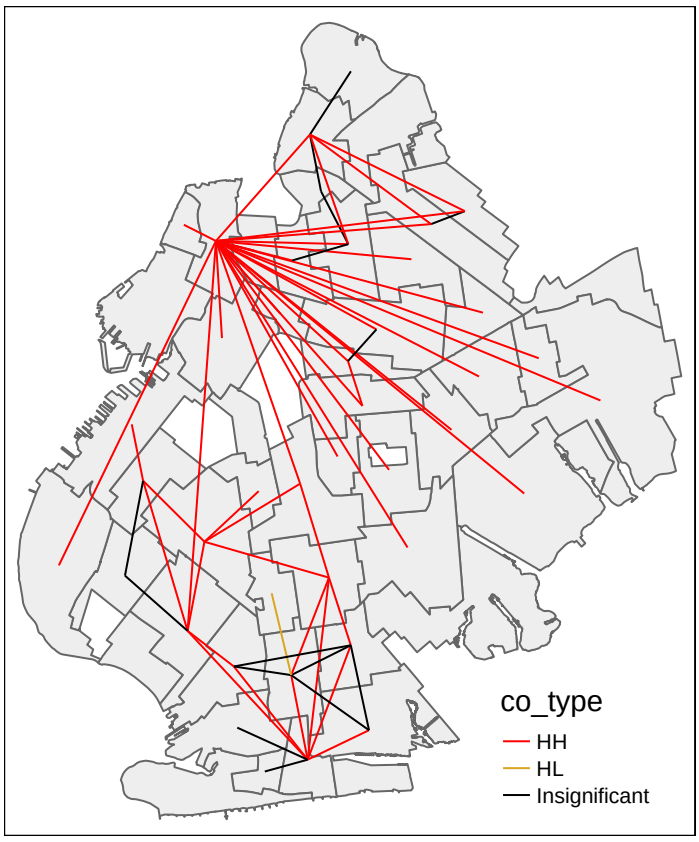

```
tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines %>% filter(co_type == "LH")) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("steelblue")
  )
```



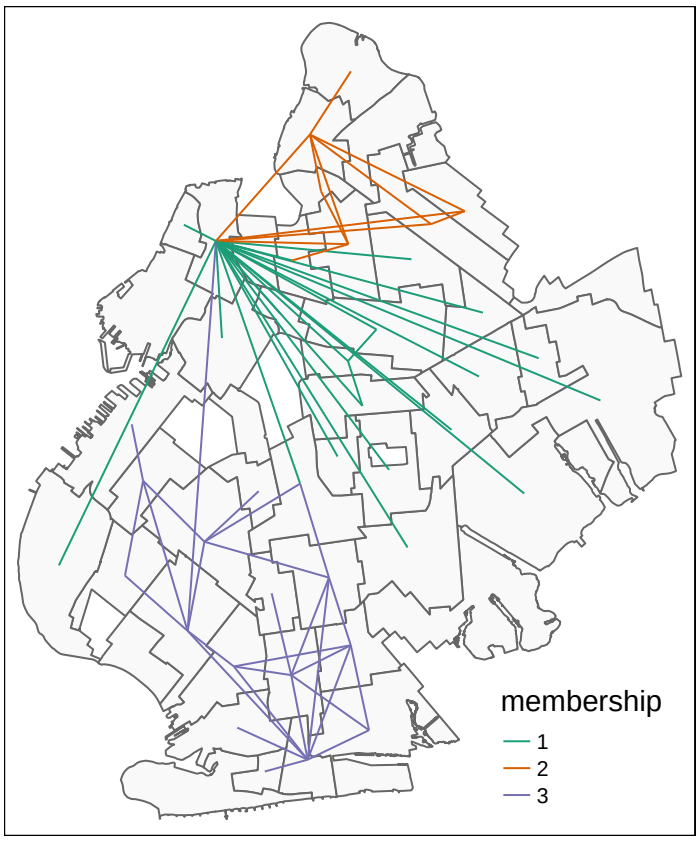

```
tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines %>% filter(co_type == "LL")) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("seagreen")
  )
```



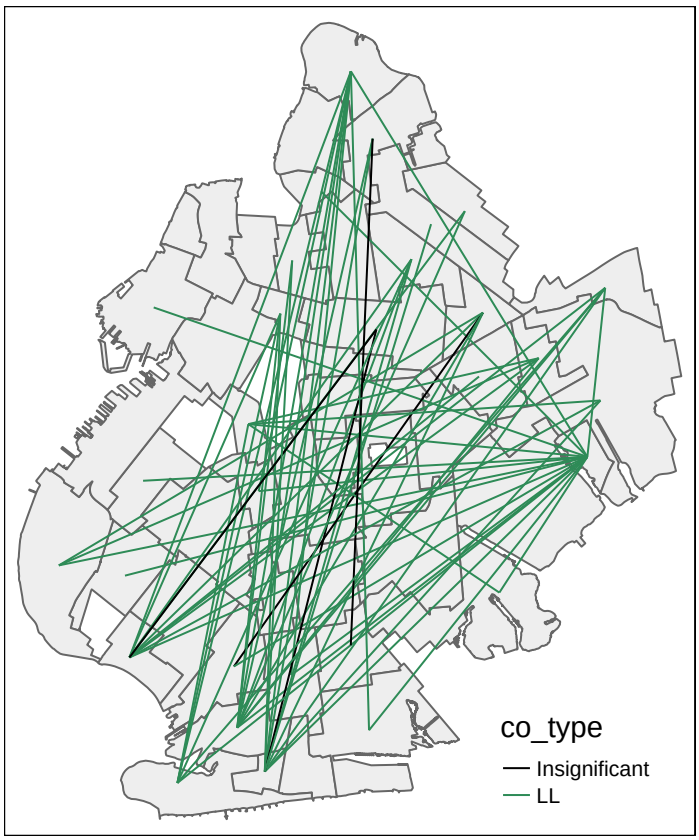

```
commute_count_lines_top <- commute_count_lines %>%
  dplyr::slice_max(
    order_by = log_S000_km2,
    n = 62
  )

 tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines_top) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("red", "goldenrod", "black")
  )
```




```
commute_network_top_clusters <- igraph::cluster_fast_greedy(commute_network_top)

commute_count_lines_top_clusters <- commute_count_lines_top %>%
  dplyr::mutate(
    membership = as.character(commute_network_top_clusters$membership)
  )

tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#f9f9f9"
  ) + tmap::tm_shape(commute_count_lines_top_clusters) +
  tmap::tm_lines(
    col = "membership",
    palette = brewer.pal(3, "Dark2")
  )
```




```
commute_count_lines_bottom <- commute_count_lines %>%
  dplyr::slice_min(
    order_by = log_S000_km2,
    n = 62
  )

 tmap::tm_shape(bk_nta_border) +
  tmap::tm_polygons(
    col = "#eeeeee"
  ) + tmap::tm_shape(commute_count_lines_bottom) +
  tmap::tm_lines(
    col = "co_type",
    palette = c("black", "seagreen")
  )
```


co_type
Insignificant
LL