Istanbul Public Transportation Network Analysis

Ahmet Emre Aladag

10 July 2015

Unweighted Network

Loading Data

Let's load libraries, stop and line data.

```
stops = read.csv("../crawler/output/stop.csv")
stops = unique(stops)
lines = read.csv("../crawler/output/linedetail.csv")
```

Edge Construction

Now create edges by connecting adjacent stops in each line:

```
newedges = list()
k = 1
for (i in 1:nrow(lines)) {
  line_name = as.character(lines$cdk_id[i])
  line_stops = strsplit(as.character(lines$stop_list[i]), ";")[[1]]
  for (j in 1:(length(line_stops)-1)){
    newedges[[k]] = c(line_stops[j], line_stops[j+1], line_name)
    k = k + 1
  }
}
```

Then we can convert this edge list into a matrix with columns (V1, V2, Line):

```
matrix_edges = matrix(data=NA, nrow=length(newedges), ncol=3)

for (i in 1:length(newedges)){
  matrix_edges[i, 1] = newedges[[i]][1]
  matrix_edges[i, 2] = newedges[[i]][2]
  matrix_edges[i, 3] = newedges[[i]][3]
}
```

Now we convert this matrix into a data frame with columns V1 (From), V2 (To) and Line (Edge label).

```
frame_edges = as.data.frame(matrix_edges, stringsAsFactors = TRUE, row.names=c("from", "to", "line"))
colnames(frame_edges) = c("V1","V2","Line")
```

Constructing Graph

Now that we have edges and nodes, we can construct a graph object.

```
library("igraph")
g = graph.data.frame(frame_edges, TRUE, stops$cdk_id)
V(g)$Latitude = stops$lat
V(g)$Longitude = stops$lon
V(g)$size = 0.5
V(g)$label = as.character(stops$name)
V(g)$labels = as.character(stops$name)
E(g)$label = as.character(frame_edges$Line)
```

Drawing the Map

Now create an aerial map:

```
library("ggmap")
```

```
## Loading required package: ggplot2
```

```
location = c(mean(stops$lon), mean(stops$lat))
map = get_map(location, maptype="watercolor", source="osm", zoom=10)
```

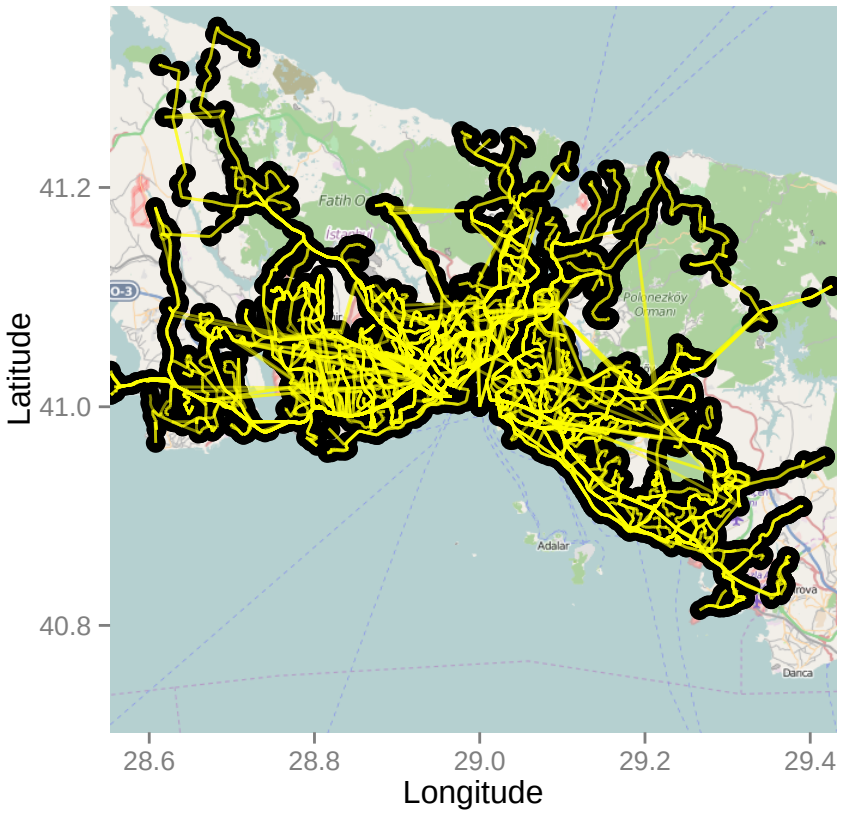
```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=41.034869,28.991441&zoom=10&size=
```

Now draw the spatial map and plot nodes/edges on it:

```
library(popgraph)
p = ggmap(map)
p = p + geom_nodeset( aes(x=Longitude, y=Latitude, size=size, label=label), g)
p = p + geom_edgeset( aes(x=Longitude, y=Latitude, shape=label, label=label), g, color="yellow", alpha=0
p + xlab("Longitude") + ylab("Latitude")
```

```
## Warning: Removed 424 rows containing missing values (geom_point).
```

```
## Warning: Removed 1453 rows containing missing values (geom_segment).
```




```
# ggsave ("map.png", dpi = 200)
```

Weighted Network

Construct the graph

We construct the graph and add an additional weight parameter to edges. Edges will be 0-1 normalized log of the frequencies (number of buses pass through that edge).

```
library(plyr)
library(ppls)
```

```
## Loading required package: splines
## Loading required package: MASS
```

```
range01 <- function(x){(x-min(x))/(max(x)-min(x))}
freqs = count(frame_edges, vars=c("V1","V2"))
g2 = graph.data.frame(freqs, TRUE, stops$cdk_id)
V(g2)$Latitude = stops$lat
V(g2)$Longitude = stops$lon
V(g2)$size = 0.5
V(g2)$label = as.character(stops$name)
V(g2)$labels = as.character(stops$name)
E(g2)$freq = as.numeric(freqs$freq)
```

```
E(g2)$weight = as.numeric(range01(log(as.numeric(freqs$freq))))
E(g2)$betweenness = edge.betweenness.estimate(g2, directed=TRUE, cutoff=10, weights = E(g2)$freq)
E(g2)$logbetweenness = log(E(g2)$betweenness)
```

Draw the map

Now we draw the map with the edge thickness and color adjusted to edge weight (bus frequency).

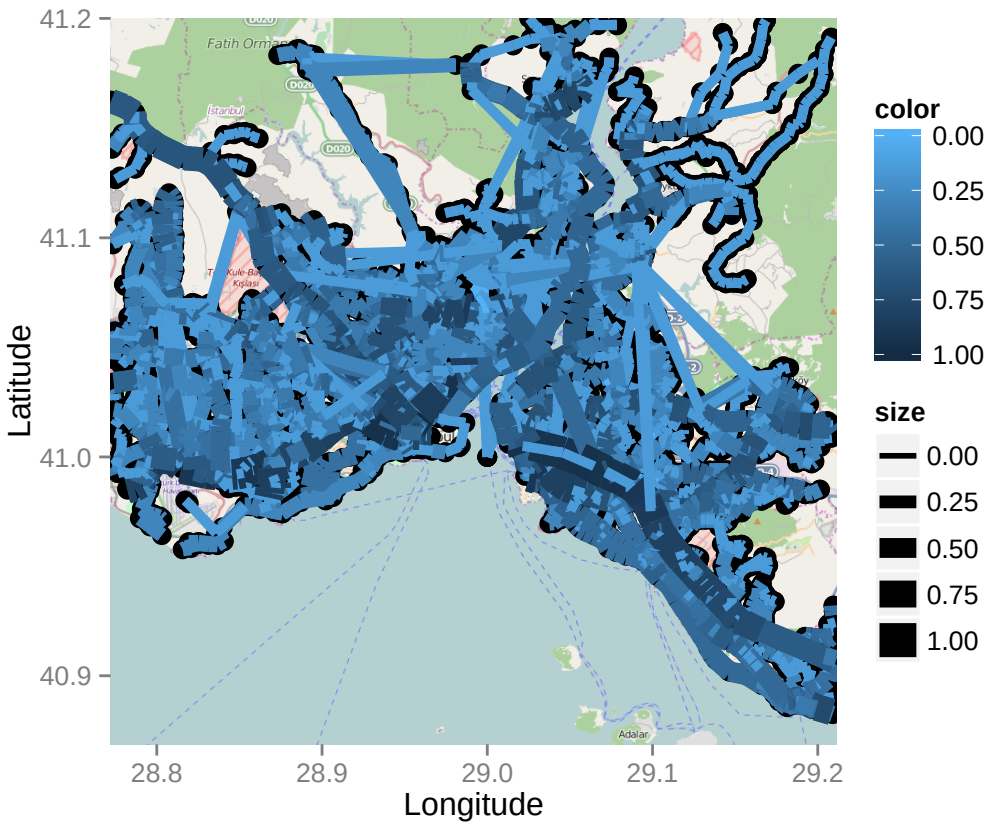
```
location = c(mean(stops$lon), mean(stops$lat))
map11 = get_map(location, maptype="watercolor", source="osm", zoom=11)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=41.034869,28.991441&zoom=11&size=
```

```
p2 = ggmap(map11)
p2 = p2 + geom_nodeset( aes(x=Longitude, y=Latitude, size=size, label=label), g2)
p2 = p2 + geom_edgeset( aes(x=Longitude, y=Latitude, shape=label, label=label, size=weight, color=weigh
p2 + xlab("Longitude") + ylab("Latitude")
```

```
## Warning: Removed 3107 rows containing missing values (geom_point).
```

```
## Warning: Removed 3728 rows containing missing values (geom_segment).
```

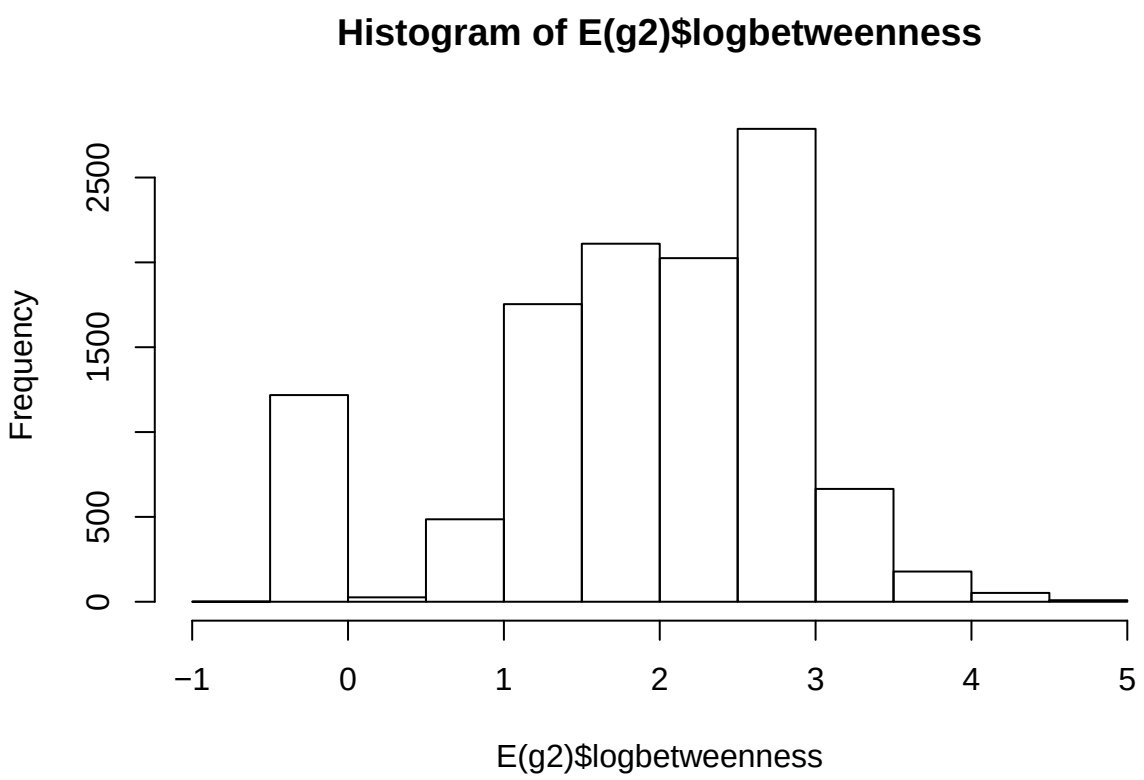


```
#ggsave ("map2.png", dpi = 200)
```

Edge Betweenness

Lets see the histogram of the edge betweenness values that are greater than 1:

```
hist(E(g2)$logbetweenness)
```



Wee see that busiest edges are ones above logbetweenness 3. So remove less busy ones.

```
small_edges = subset(E(g2), E(g2)$logbetweenness < 3)
g3 = delete.edges(g2, small_edges)
```

Now plot the map according to logarithmic edge betweenness:

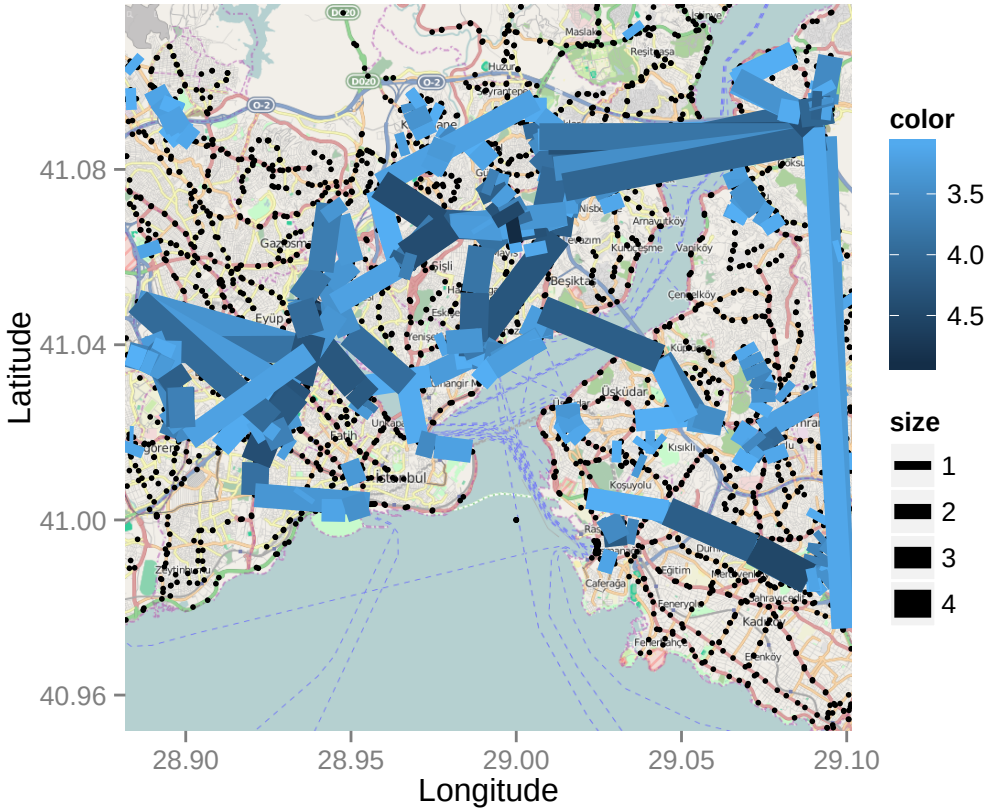
```
location = c(mean(stops$lon), mean(stops$lat))
map12 = get_map(location, maptype="watercolor", source="osm", zoom=12)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=41.034869,28.991441&zoom=12&size=
```

```
p3 = ggmap(map12)
p3 = p3 + geom_nodeset( aes(x=Longitude, y=Latitude, size=size, label=label), g3)
p3 = p3 + geom_edgeset( aes(x=Longitude, y=Latitude, shape=label, label=label, size=logbetweenness, col
p3 + xlab("Longitude") + ylab("Latitude")
```

```
## Warning: Removed 6343 rows containing missing values (geom_point).
```

```
## Warning: Removed 573 rows containing missing values (geom_segment).
```




```
#ggsave ("map3.png", dpi = 200)
```

We can see that two bridges of Istanbul have the highest betweenness values. This means that they have very important roles in Istanbul transportation and a failure in the service would cause lots of delays. Next, we can see roads Besiktas-Levent, Sisli-Kagithane, around Sisli, Halic Bridge, Yenisahra E-5. Then we can see the highway (on the rightmost vertical) connecting E-5 to north bridge.

Node Betweenness

Calculate node betweenness and sort nodes by that:

```
V(g2)$betweenness = betweenness(g2, v=V(g2), directed = TRUE, weights = E(g2)$freq)
V(g2)$logbetweenness = log(V(g2)$betweenness)
dd = get.data.frame(g2, what=c("vertices"))

sorted_list = dd[order(dd$betweenness, decreasing = TRUE),]

for (i in 1:20 ) {
  print(sorted_list$label[i])
}
```

```
## [1] "SAB\304\260HA G\303\226K\303\207EN \304\260.D.H."
## [1] "KAVACIK K\303\226PR\303\234S\303\234"
```

```
## [1] "TAKS\304\260M"
## [1] "SAB\304\260HA G\303\226K\303\207EN N\304\260Z."
## [1] "EM\304\260N\303\226N\303\234 \304\260SKELE"
## [1] "YEN\304\260KAPI SAH\304\260L"
## [1] "KUMKAPI"
## [1] "\303\207ATLADIKAPI"
## [1] "AKBIYIK"
## [1] "SARAYBURNU"
## [1] "SALIPAZARI"
## [1] "TEKN\304\260K \303\234N\304\260VERS\304\260TE"
## [1] "MEVLANAKAPI"
## [1] "S\304\260L\304\260VR\304\260KAPI"
## [1] "DOLMABAH\303\207E SARAYI"
## [1] "M\304\260N\304\260AT\303\234RK"
## [1] "HAL\304\260\303\207 KONGRE MERKEZ\304\260"
## [1] "KAVACIK K\303\226PR\303\234S\303\234"
## [1] "4.LEVEND"
## [1] "TOPKAPI ALT GE\303\207\304\260T"
```

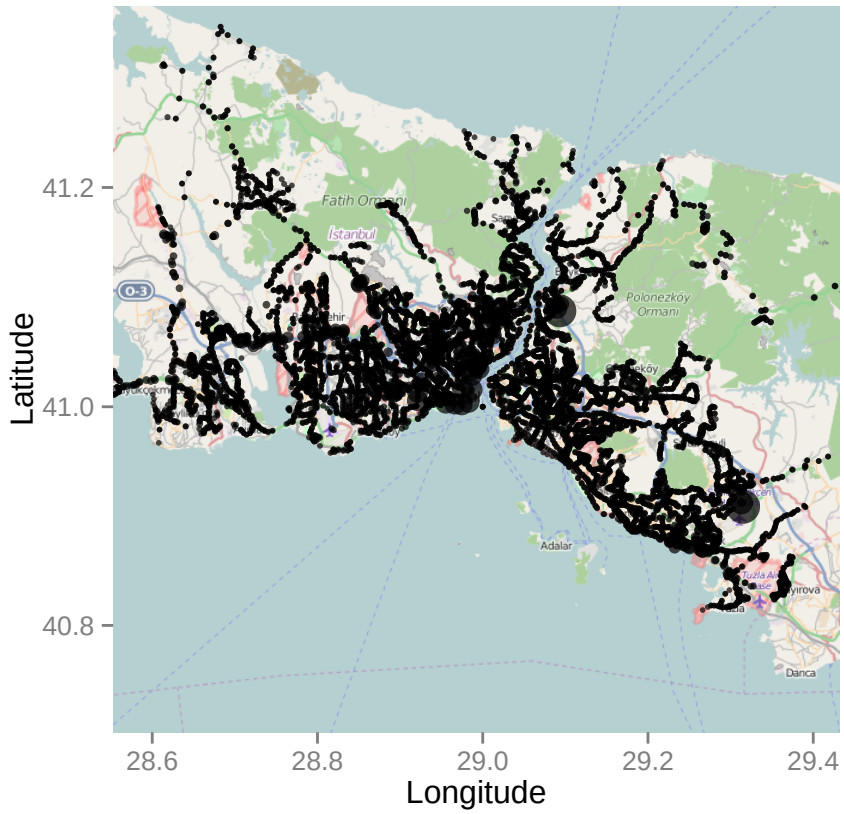
Now plot the map according to node betweenness:

```
location = c(mean(stops$lon), mean(stops$lat))
map10 = get_map(location, maptype="watercolor", source="osm", zoom=10)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=41.034869,28.991441&zoom=10&size=
```

```
p4 = ggmap(map10)
p4 = p4 + geom_nodeset( aes(x=Longitude, y=Latitude, size=betweenness, label=label), g2, alpha=0.7)
p4 + xlab("Longitude") + ylab("Latitude")
```

```
## Warning: Removed 424 rows containing missing values (geom_point).
```




```
#ggsave ("map4.png", dpi = 200)
```

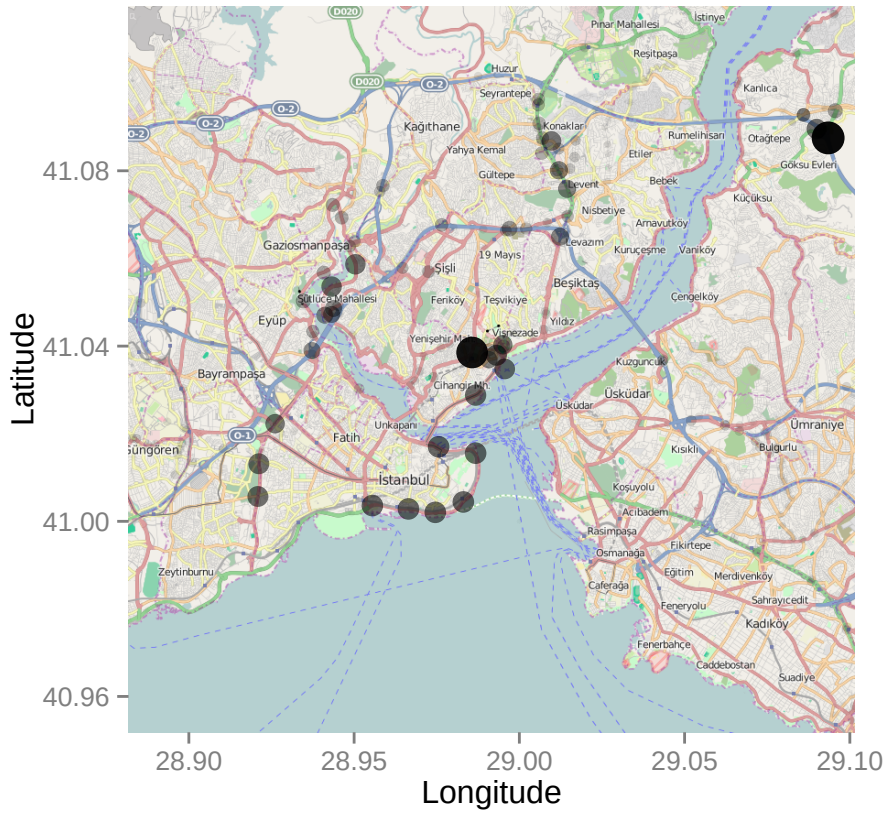
This time adjust alpha with 0-1 normalized node betweenness to see major ones. See in a more detailed map:

```
location = c(mean(stops$lon), mean(stops$lat))
map12 = get_map(location, maptype="watercolor", source="osm", zoom=12)
```

```
## Map from URL : http://maps.googleapis.com/maps/api/staticmap?center=41.034869,28.991441&zoom=12&size=
```

```
p5 = ggmap(map12)
p5 = p5 + geom_nodeset( aes(x=Longitude, y=Latitude, size=betweenness, label=label), g2, alpha=range01(
p5 + xlab("Longitude") + ylab("Latitude")
```

```
## Warning: Removed 6343 rows containing missing values (geom_point).
```




```
#ggsave ("map5.png", dpi = 200)
```