HW 5

Joe Brew

10/05/2014

The code for the production of this document is available at https://github.com/joebrew/uf.

Task 1

For each family address, please find the nearest EMS and calculate the straight-line distance. List the family address ID and EMS ID and the nearest distance below.

(If you use the "Near" tool, please listthe family address ID "OBJECTID," the EMS ID "NEAR_FID," and the nearest distance "NEAR_DIST."" If you use "Generate Near Table," Please list "IN_FID," "NEAR_FID" and "NEAR_DIST").

Set working directory:

```
if ( Sys.info()["sysname"] == "Linux" ){
  setwd("/home/joebrew/")
} else {
  setwd("C:/Users/BrewJR/")
}

mywd <- paste0(getwd(), "/Documents/uf/phc6194/hw5")
setwd(mywd)
```

Once could read in the mdb file using the Hmisc library and mdb-tools.

```
# Attach the necessary package
library(Hmisc)

# View which tables are available
mdb.get("Homwork5.mdb", tables = TRUE)

# Read in the Family_address table
fam <- mdb.get("Homwork5.mdb", tables = "Family_address")
gdb <- mdb.get("Homwork5.mdb", tables = "GDB_Items")
road <- mdb.get("Homwork5.mdb", tables = "road")
```

However, the above method doesn't (easily) keep the spatial aspects associated with each table. Insead, I used ArcGIS to read in the .mdb database, and then exported the road, EMS and addresses shapefiles seperately.

```
library(rgdal)
road <- readOGR(".", "road")
ems <- readOGR(".", "EMS_LOCATION")
fam <- readOGR(".", "Family_address")
```

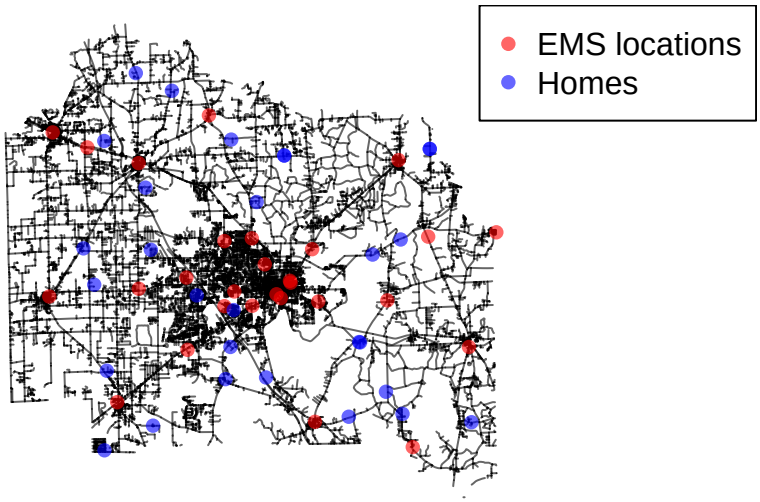
Check that our projection and coordinate systems are identical:

```
proj4string(ems) == proj4string(fam)
```

```
## [1] TRUE
```

Confirm that we read everything in okay by plotting our map and points

```
plot(road, col = adjustcolor("black", alpha.f = 0.6))
points(ems, col = adjustcolor("red", alpha.f = 0.6), pch = 16)
points(fam, col = adjustcolor("blue", alpha.f = 0.6), pch = 16)
legend("topright", pch = 16, col = adjustcolor(c("red", "blue"), alpha.f = 0.6),
       legend = c("EMS locations", "Homes"))
```



Check what kind of unit of measurement we're working with

```
proj4string(ems) # US-FT
```

```
## [1] "+proj=lcc +lat_1=29.58333333333333 +lat_2=30.75 +lat_0=29 +lon_0=-84.5 +x_0=600000 +y_0=0 +datu
```

Merge our EMS and family points into one dataframe.

```
fam_coords <- data.frame(cbind(coordinates(fam), group = "fam"), stringsAsFactors = FALSE)
ems_coords <- data.frame(cbind(coordinates(ems), group = "ems"), stringsAsFactors = FALSE)

# Create a list of the two collections of coordinates
x <- list(ems_coords, fam_coords)
# Bind all elementsin the list together
x <- do.call("rbind", x)
# Now we have x, a dataframe of the coordinates

# Make it spatial
x$coords.x1 <- as.numeric(x$coords.x1)
x$coords.x2 <- as.numeric(x$coords.x2)
coordinates(x) <- ~ coords.x1 + coords.x2
```

Replace ems/fam_coords with their spatial equivalents

```
ems_coords <- x[which(x$group == "ems"),]
fam_coords <- x[which(x$group == "fam"),]
```

Write a function for calculating distance (in feet) using pythagorean theorem

```
DistFun <- function(x, y, x2, y2){
  xdist <- sqrt((x - x2)^2)
  ydist <- sqrt((y - y2)^2)

  linedist <- sqrt((xdist^2) + (ydist^2))
  return(linedist)
}
```

Now loop that function over each address fam_coords to extract the index of the closest ems

```
fam_coords$closest_ems_name <- vector(mode = "numeric", length = nrow(fam_coords))
fam_coords$closest_ems_distance <- vector(mode = "numeric", length = nrow(fam_coords))
fam_coords$closest_ems_index <- vector(mode = "numeric", length = nrow(fam_coords))

for (i in 1:nrow(fam_coords)){
  # Get the distance of every ems station from every address
  temp <- DistFun(x = coordinates(fam_coords)[i,1],
        y = coordinates(fam_coords)[i,2],
        x2 = coordinates(ems)[,1],
        y2 = coordinates(ems)[,2])
  # Extract the index number of the closest ems station
  best.ind <- which.min(temp)

  #Assign that index and distance to fam_coords
  fam_coords$closest_ems_index[i] <- best.ind
  fam_coords$closest_ems_name[i] <- as.character(ems$DESCRIPT[best.ind])
  fam_coords$closest_ems_distance[i] <- temp[which.min(temp)]

}

# Just for fun, let's make a miles column as well
fam_coords$closest_ems_miles <- fam_coords$closest_ems_distance / 5280

# Throw back in the object id
fam_coords$OBJECTID <- 1:nrow(fam_coords)
fam_coords$address <- fam$FULLADDR
```

Show the table with the closest EMS stations for each house.

```
library(knitr)
z <- data.frame(fam_coords[,c("OBJECTID",
                          "address", "closest_ems_name",
                          "closest_ems_miles",
                          "closest_ems_index")])
# print(xtable(z), type = "html")
kable(z, format = "markdown", row.names = NA)
```

	OBJECTID	address	closest_ems_name	closest_e
29	1	25702 NW COUNTY RD 241	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
30	2	23521 NW COUNTY RD 239	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
31	3	18028 NW 177TH AV	ALACHUA COUNTY FIRE RESCUE STATION 20	
32	4	20972 NW 46TH AV	NEWBERRY FIRE RESCUE STATION 28	
33	5	1407 NW 202ND ST	ALACHUA COUNTY FIRE RESCUE STATION 17	
34	6	10860 NE STATE RD 26	WINDSOR FIRE RESCUE	
35	7	14213 NE STATE RD 26	ALACHUA COUNTY FIRE RESCUE STATION 25	
36	8	1020 NE 156TH AV	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
37	9	1028 NE 156TH AV	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
38	10	3411 NW 177TH AV	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
39	11	1815 NW 102ND PL	ALACHUA COUNTY FIRE RESCUE STATION 9	
40	12	6410 SE 92ND TER	WINDSOR FIRE RESCUE	
41	13	6415 SE 92ND TER	WINDSOR FIRE RESCUE	
42	14	12318 S COUNTY RD 325	ALACHUA COUNTY FIRE RESCUE STATION 31	
43	15	7520 SE COUNTY RD 346	MICANOPY FIRE DEPARTMENT STATION 26	
44	16	10530 SW 12TH TER	MICANOPY FIRE DEPARTMENT STATION 26	
45	17	4350 SW WACAHOOTA RD	ALACHUA COUNTY FIRE RESCUE STATION 15	
46	18	4042 SW 69TH AV	ALACHUA COUNTY FIRE RESCUE STATION 19	
47	19	2228 SW 37TH ST	ALACHUA COUNTY FIRE RESCUE STATION 19	
48	20	7609 SW 4TH PL	ALACHUA COUNTY FIRE RESCUE STATION 16	
49	21	4630 NW 129TH ST	ALACHUA COUNTY FIRE RESCUE STATION 17	
50	22	12111 NW 136TH ST	ALACHUA COUNTY FIRE AND RESCUE STATION 21	
51	23	12930 SW 159TH AV	ALACHUA COUNTY FIRE RESCUE	
52	24	18908 SW 186TH ST	ALACHUA COUNTY FIRE RESCUE	
53	25	18318 SW 95TH AV	ALACHUA COUNTY FIRE RESCUE	
54	26	16110 NE COUNTY RD 1471	WALDO FIRE AND RESCUE STATION 23	
55	27	16010 NE COUNTY RD 1471	WALDO FIRE AND RESCUE STATION 23	
56	28	22240 SE 162ND AV	ALACHUA COUNTY FIRE RESCUE STATION 31	
57	29	13909 SE 152ND LN	ALACHUA COUNTY FIRE RESCUE STATION 31	

Task 2

Use network analysis to estimate the shortest response time of an EMS (emergence medical service) to the family addresses following the in- class tutorials.

Steps: 1. Data Preparation: Assign travel time to road segment following the in- class tutorials; 2. Build Network Dataset; 3. Estimate shortest travel time 4. Note: Select “new OD Cost Matrix” from Network

Analyst and Select "Family_house" as Origins and "EMS_Location" as Destinations. Please give a good screen shot to show your result.

I took an alternative approach to this problem. Using the ggmap package, I calculated the *true* travel time from every address to every EMS location. The advantage of this approach is that it takes into account travel times from Google's API, rather than simply calculating speed limit (which likely underestimates true travel time).

First, I attach the ggmap library.

```
library(ggmap)
```

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

Now, let's get latitude and longitude for each point (so as to more smoothly interact with google's API):

```
#Get a projection string for both
proj4string(ems_coords) <- proj4string(ems)
proj4string(fam_coords) <- proj4string(ems)

#Convert to latitude / longitude
ems_latlon <- spTransform(ems_coords, CRS("+init=epsg:4326"))
fam_latlon <- spTransform(fam_coords, CRS("+init=epsg:4326"))
```

Then I write a function to wrap around ggmap's mapdist() function

```
TravelTime <- function(from, to){
  mapdist(from, to, mode = "driving")
}
```

Finally, I loop my TravelTime function over each row of the address points.

```
# Create empty columns for our new variables
fam_coords$closest_ems_travel_time_minutes <- vector(mode = "numeric", length = nrow(fam_coords))
fam_coords$closest_ems_travel_distance_km <- vector(mode = "numeric", length = nrow(fam_coords))
fam_coords$closest_ems_travel_index <- vector(mode = "numeric", length = nrow(fam_coords))
fam_coords$closest_ems_travel_name <- vector(mode = "numeric", length = nrow(fam_coords))

#Loop through each row of family addresses, calculating distance/time from each ems
for (i in 1:nrow(fam_coords)){
  tofam <- paste0(coordinates(fam_latlon)[,2], ", ",coordinates(fam_latlon)[,1])[i]
  fromems <- paste0(coordinates(ems_latlon)[,2], ", ",coordinates(ems_latlon)[,1])

  x <- TravelTime(from = fromems,
             to = tofam)

 # Assing the index of the best (least time) ems station
  best.ind <- which.min(x$minutes)[1]

 # Put info back into fam_coords
fam_coords$closest_ems_travel_time_minutes[i] <- x$minutes[best.ind]
fam_coords$closest_ems_travel_distance_km[i] <- x$km[best.ind]
fam_coords$closest_ems_travel_index[i] <- best.ind
```

```
fam_coords$closest_ems_travel_name[i] <- as.character(ems$DESCRIPT[best.ind])

}
```

Show the table with the closest EMS stations for each house.

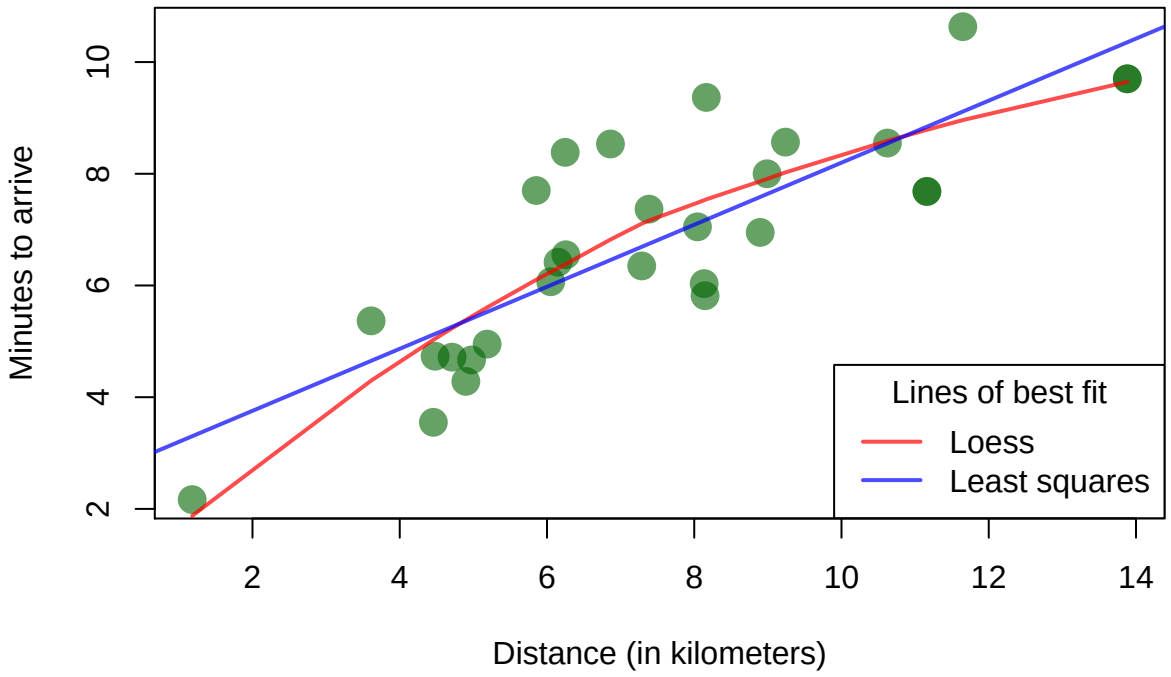
```
library(knitr)
z <- data.frame(fam_coords[,c("OBJECTID",
                          "address", "closest_ems_travel_name",
                          "closest_ems_travel_distance_km",
                          "closest_ems_travel_time_minutes",
                          "closest_ems_index")])
# print(xtable(z), type = "html")
kable(z, format = "markdown", row.names = NA)
```

	OBJECTID	address	closest_ems_travel_name	closest_e
29	1	25702 NW COUNTY RD 241	ALACHUA COUNTY FIRE AND RESCUE STATION 21	
30	2	23521 NW COUNTY RD 239	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
31	3	18028 NW 177TH AV	ALACHUA COUNTY FIRE RESCUE STATION 20	
32	4	20972 NW 46TH AV	NEWBERRY FIRE RESCUE STATION 28	
33	5	1407 NW 202ND ST	ALACHUA COUNTY FIRE RESCUE STATION 17	
34	6	10860 NE STATE RD 26	ALACHUA COUNTY FIRE RESCUE STATION 25	
35	7	14213 NE STATE RD 26	ALACHUA COUNTY FIRE RESCUE STATION 25	
36	8	1020 NE 156TH AV	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
37	9	1028 NE 156TH AV	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
38	10	3411 NW 177TH AV	LA CROSSE VOLUNTEER FIRE DEPARTMENT	
39	11	1815 NW 102ND PL	ALACHUA COUNTY FIRE RESCUE STATION 9	
40	12	6410 SE 92ND TER	WINDSOR FIRE RESCUE	
41	13	6415 SE 92ND TER	WINDSOR FIRE RESCUE	
42	14	12318 S COUNTY RD 325	ALACHUA COUNTY FIRE RESCUE STATION 31	
43	15	7520 SE COUNTY RD 346	MICANOPY FIRE DEPARTMENT STATION 26	
44	16	10530 SW 12TH TER	MICANOPY FIRE DEPARTMENT STATION 26	
45	17	4350 SW WACAHOOTA RD	ALACHUA COUNTY FIRE RESCUE STATION 15	
46	18	4042 SW 69TH AV	ALACHUA COUNTY FIRE RESCUE STATION 19	
47	19	2228 SW 37TH ST	ALACHUA COUNTY FIRE RESCUE STATION 19	
48	20	7609 SW 4TH PL	ALACHUA COUNTY FIRE RESCUE STATION 16	
49	21	4630 NW 129TH ST	ALACHUA COUNTY FIRE RESCUE STATION 17	
50	22	12111 NW 136TH ST	ALACHUA COUNTY FIRE AND RESCUE STATION 21	
51	23	12930 SW 159TH AV	ALACHUA COUNTY FIRE RESCUE	
52	24	18908 SW 186TH ST	ALACHUA COUNTY FIRE RESCUE	
53	25	18318 SW 95TH AV	ALACHUA COUNTY FIRE RESCUE	
54	26	16110 NE COUNTY RD 1471	ALACHUA COUNTY FIRE RESCUE STATION 25	

	OBJECTID	address	closest_ems_travel_name	closest_e
55	27	16010 NE COUNTY RD 1471	ALACHUA COUNTY FIRE RESCUE STATION 25	
56	28	22240 SE 162ND AV	CITY OF HAWTHORNE FIRE AND RESCUE	
57	29	13909 SE 152ND LN	ALACHUA COUNTY FIRE RESCUE STATION 31	

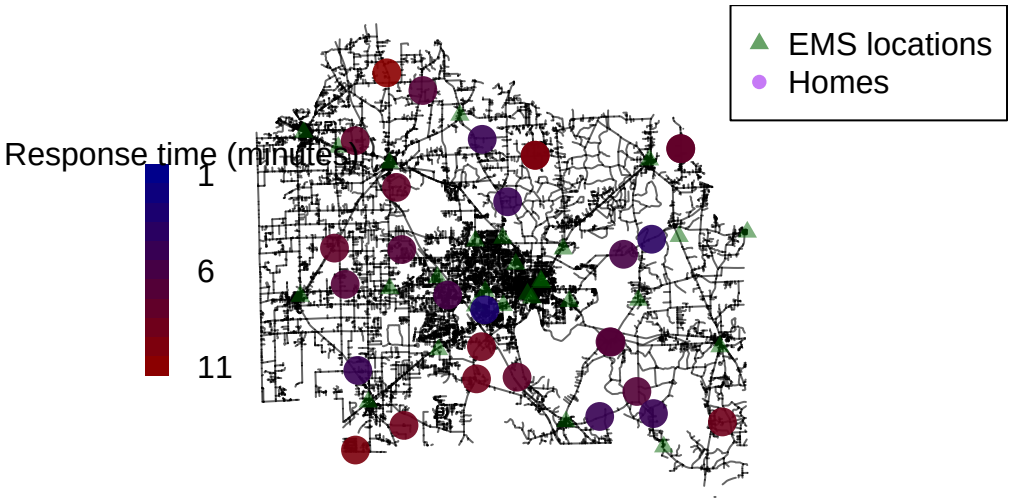
Note that there is a correlation between distance and time, but it's not perfectly linear.

```
plot(fam_coords$closest_ems_travel_distance_km,
     fam_coords$closest_ems_travel_time_minutes,
     xlab = "Distance (in kilometers)",
     ylab = "Minutes to arrive",
     pch = 16,
     col = adjustcolor("darkgreen", alpha.f = 0.6),
     cex = 2)

  #ADD LOESS LINE
  lox <- fam_coords$closest_ems_travel_distance_km
  loy <- fam_coords$closest_ems_travel_time_minutes
  lw1 <- loess(loy ~ lox, span=1)
  j <- order(lox)
  lines(lox[j],lw1$fitted[j],col=adjustcolor("red", alpha.f = 0.7),
        lty=1,
        lwd = 2)

# ADD Regression line
fit <- lm(loy ~ lox)
abline(fit, col = adjustcolor("blue", alpha.f = 0.7),
       lwd = 2)
legend("bottomright",
       col = adjustcolor(c("red", "blue"), alpha.f = 0.7),
       lty = 1,
       lwd = 2,
       legend = c("Loess", "Least squares"),
       title = "Lines of best fit")
```



Just for fun, let's plot the family addresses with their color being a function of the time they are from an EMS response (red being slow, blue being fast).

```
library(RColorBrewer)
my_colors <- colorRampPalette(c("darkblue", "darkred"))(ceiling(max(fam_coords$closest_ems_travel_time_m
fam_coords$color <- my_colors[ceiling(fam_coords$closest_ems_travel_time_minutes)]

plot(road, col = adjustcolor("black", alpha.f = 0.6))
points(ems, col = adjustcolor("darkgreen", alpha.f = 0.5), pch = 17)
points(fam, col = adjustcolor(fam_coords$color, alpha.f = 0.9), pch = 16, cex = 2)
legend("topright", pch = c(17,16), col = adjustcolor(c("darkgreen", "purple"), alpha.f = 0.6),
       legend = c("EMS locations", "Homes"))
legend("left", fill = my_colors, bty = "n",
       y.intersp = 0.5, ncol=1, legend=c(1, rep(NA, 4), 6, rep(NA, 4), 11),
       border = NA, title = "Response time (minutes)")
```



Here's the code for an interactive map (if you run this on your own computer)

```
library(rCharts)

# read in geojson version of zip
#zip_geoj <- readOGR("zips_alachua", "ACDPS_zipcode")
mymap <- Leaflet$new()
mymap$tileLayer(provider = "Stamen.TonerLite")
mymap$setView(c(29.65, -82.3), zoom = 10)
mymap$enablePopover(TRUE)


library(rCharts)
library(rMaps)
mywaypoints = list(c(40.74119, -73.9925), c(40.73573, -73.99302))


mywaypoints <- list()
for (i in 1:nrow(fam_coords)){
  mywaypoints[[i]] <- (c(as.numeric(coordinates(fam_latlon)[i,2]),
                as.numeric(coordinates(fam_latlon)[i,1])))

}


mymap$addAssets(
  css = "http://www.liedman.net/leaflet-routing-machine/dist/leaflet-routing-machine.css",
  jshead = "http://www.liedman.net/leaflet-routing-machine/dist/leaflet-routing-machine.min.js"
)

routingTemplate = "
 <script>
 var mywaypoints = %s
 L.Routing.control({
  waypoints: [
    L.latLng.apply(null, mywaypoints[0]),
    L.latLng.apply(null, mywaypoints[1])
  ]
 }).addTo(mymap);
 </script>"

mymap$setTemplate(
  afterScript = sprintf(routingTemplate, RJSONIO::toJSON(mywaypoints))
)
mymap$set(width = 1450, height = 800)

mymap

#mymap$fullScreen(TRUE)

# for (i in 1:nrow(fam_coords)){
#   mymap$marker(c(as.numeric(coordinates(fam_latlon)[i,2]),
#                  as.numeric(coordinates(fam_latlon)[i,1])),
#                bindPopup = paste(round(fam_coords$closest_ems_travel_time_minutes[i], digits=1), "min
```

```
#                                   fam_coords$closest_ems_travel_name[i], "to respond to",
#                                   fam_coords$address[i]
#                                   ))
#
# }

#library(shiny)
# renderMap(mymap)
```