Spatial Modeling, Data, and Statistics in R

Amy Wesolowski (awesolowski@jhu.edu (mailto:awesolowski@jhu.edu))

```

First, you need to set your current working directory to the location where the spatial data files are located. This will be different for each person.

```
setwd('~/Dropbox/Teaching/SpatialStatsMada/')
```

# Reading shp files

For the first part of the lecture, we will go through different spatial data sets and how you can load them into R. There are multiple packages that will enable you to read in .shp files in R, but here we will use maptools. It is important that all of the corresponding files (.dbf, .prj, .shx) are in the same folder as the .shp file. These other files include additional information about the spatial data frame that are necessarily to read a .shp file. We will use two functions, one to read in a point shp file and one to read in a polygon shp file. If an error message comes up that says use rgdal::readOGR or sf::st_read, these messages are just to indicate that other functions to read shp files may be more efficient or newer. First we will read in

```
library(maptools)
```

```
## Loading required package: sp
```

```
## Checking rgeos availability: TRUE
```

```
library(rgdal)
```

```
## rgdal: version: 1.2-8, (SVN revision 663)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.1.3, released 2017/20/01
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.4/Resources
/library/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.3, 15 August 2016, [PJ_VERSION: 493]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.4/Resourc
es/library/rgdal/proj
##  Linking to sp version: 1.2-4
```
```

```
library(rgeos)
```

```
## rgeos version: 0.3-23, (SVN revision 546)
##  GEOS runtime version: 3.6.1-CAPI-1.10.1 r0
##  Linking to sp version: 1.2-4
##  Polygon checking: TRUE
```

```
mdg_admin2_shp<-readShapePoly('MDG_Shp/MDG_adm2.shp', proj4string = CRS('+proj=longla
t'))
```

```
## Warning: use rgdal::readOGR or sf::st_read
```

```
mdg_admin2_shp<-gSimplify(mdg_admin2_shp, tol = 0.01)

par(mfrow=c(1,3))
plot(mdg_admin2_shp)
```

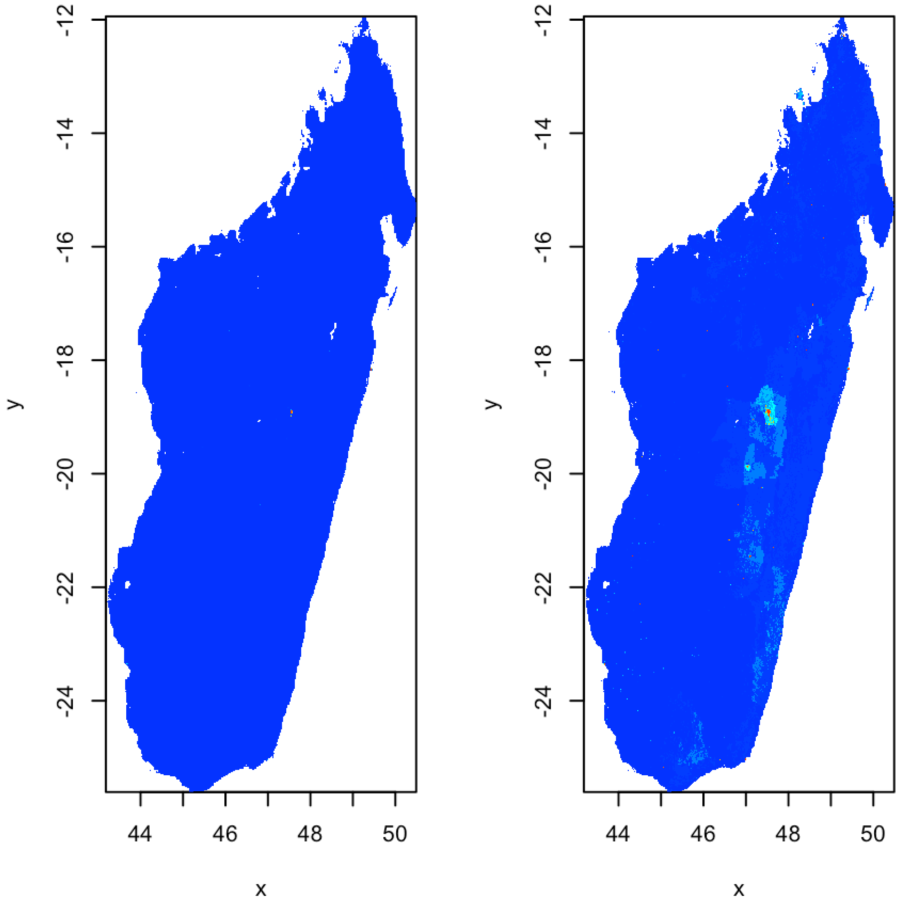
Reading in Raster Data sets

Next we will practice reading in raster data and extracting these data

```
library(raster)
library(rgdal)
library(colorRamps)

mdg_birth<-raster('MDG_Births/MDG_births_pp_v2_2015.tif')
```

After reading in these data, we will then plot these raster images using various color schemes and highlighting particular values.

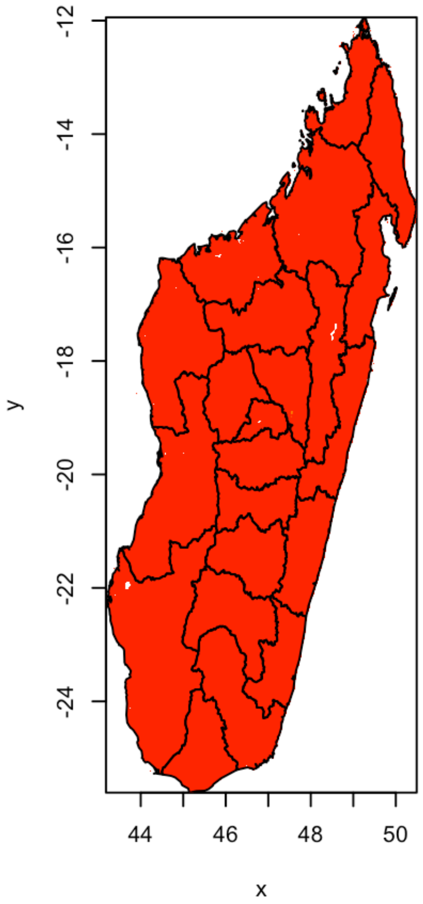
```
par(mfrow=c(1,3))
image(mdg_birth, col = blue2red(10))
image(log(mdg_birth+1), col = blue2red(10))
```



- TO DO: How does the image change if you change the raster values to be between 0 and 10, or 50 and above?

Now we will also plot these raster images with the Madagascar shp files

```
par(mfrow=c(1,3))
image(mdg_birth, zlim = c(0,3000))
plot(mdg_admin2_shp, add = TRUE)
```



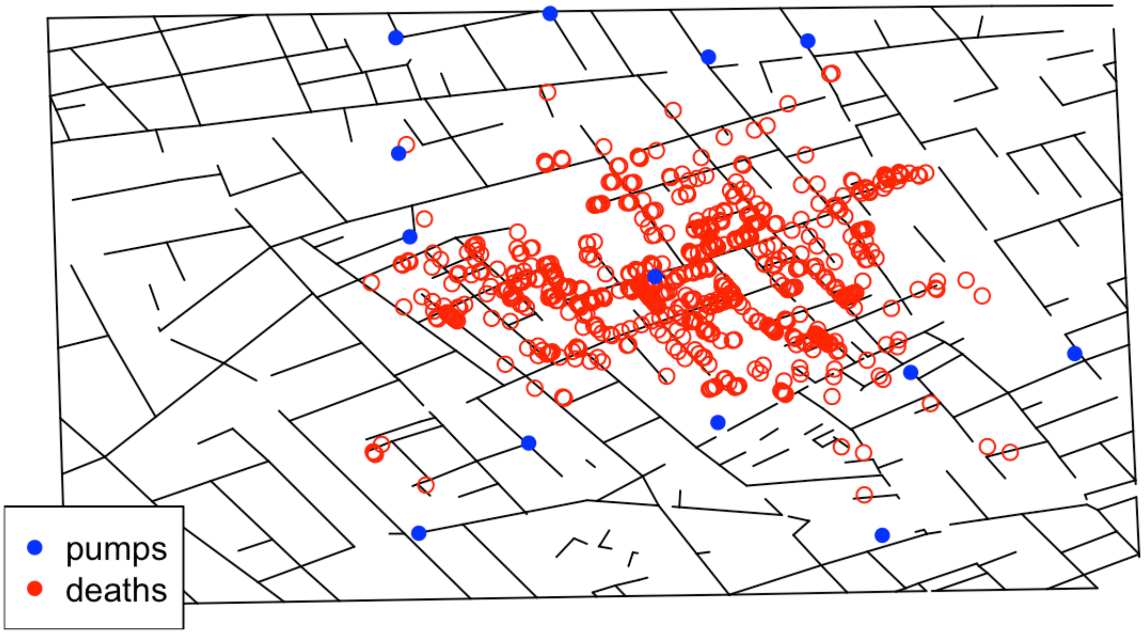
Reading in point data sets

We will work with the Snow data again and calculate some additional summary values. The pumps and deaths datapoints were digitized by Rusty Dodson at the National Center for Geographic Information & Analysis (NCGIA) at UC Santa Barbara. First we will read in each data set.

```
#library(spatstat) # do not need, but has lots of great functions for spatial point p
rocess data

deaths_points_file<-read.csv('Snow_deaths.csv', header = TRUE)

pumps_points_file<-read.csv('Snow_pumps.csv', header = TRUE)

street_points_file<-read.csv('Snow_streets.csv', header = TRUE)
```

The death and pump data set are x,y coordinates (not longitude/latitude) that can be read and plot directly. The street data set is read in as endpoints of lines that will need to be plot in a different manner. First we will remake the plot that we made during the lecture.

```
par(mfrow=c(1,1))
plot(NA, NA, xlim = range(street_points_file$x), ylim = range(street_points_file$y),
xlab = '', ylab = '', bty = 'n', xaxt = 'n', yaxt = 'n')

for(ii in 1:max(street_points_file$street)){
  sub_dat<-street_points_file[which(street_points_file$street == ii),]
  lines(c(sub_dat$x[1], sub_dat$x[2]), c(sub_dat$y[1], sub_dat$y[2]))
}

points(deaths_points_file$x, deaths_points_file$y, col = 'red')

points(pumps_points_file$x, pumps_points_file$y, col = 'blue', pch = 16)

legend('bottomleft', legend = c('pumps', 'deaths'), col = c('blue', 'red'), pch = 16)
```

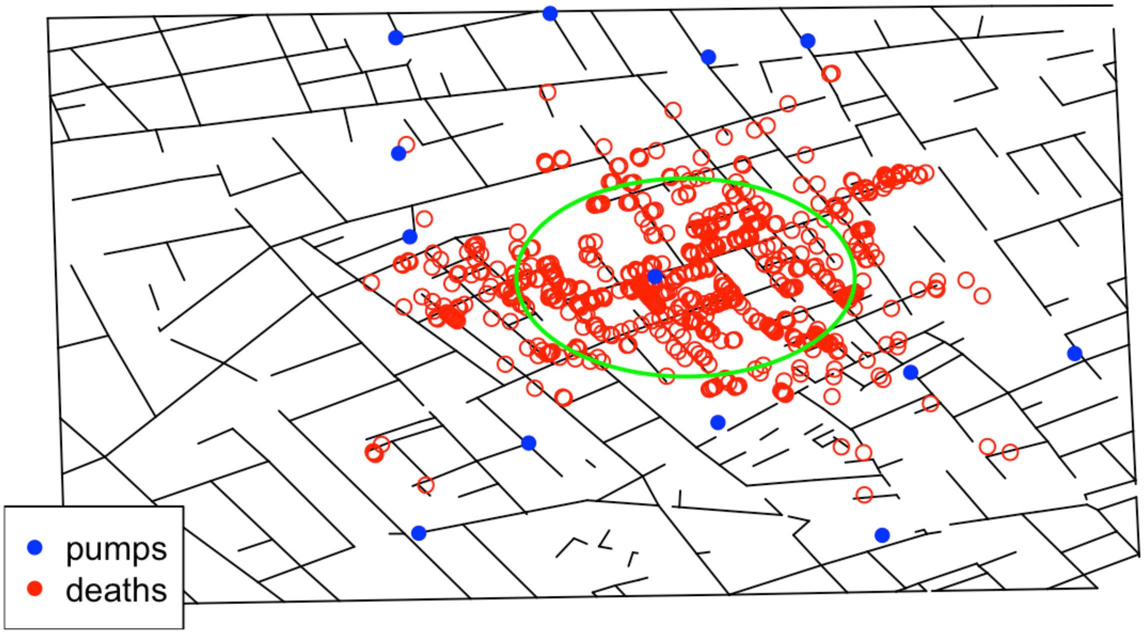


We will also write two additional functions that we will use to plot both the basic streets as well as the pump, death and street data.

```
plot_streets<-function(){
  plot(NA, NA, xlim = range(street_points_file$x), ylim = range(street_points_file$y)
, xlab = '', ylab = '', bty = 'n', xaxt = 'n', yaxt = 'n', add = TRUE)

  for(ii in 1:max(street_points_file$street)){
    sub_dat<-street_points_file[which(street_points_file$street == ii),]
    lines(c(sub_dat$x[1], sub_dat$x[2]), c(sub_dat$y[1], sub_dat$y[2]))
  }
}

plot_base_snow_plot<-function(){
  par(mfrow=c(1,1))
  plot(NA, NA, xlim = range(street_points_file$x), ylim = range(street_points_file$y)
, xlab = '', ylab = '', bty = 'n', xaxt = 'n', yaxt = 'n')

  for(ii in 1:max(street_points_file$street)){
    sub_dat<-street_points_file[which(street_points_file$street == ii),]
    lines(c(sub_dat$x[1], sub_dat$x[2]), c(sub_dat$y[1], sub_dat$y[2]))
  }

  points(deaths_points_file$x, deaths_points_file$y, col = 'red')

  points(pumps_points_file$x, pumps_points_file$y, col = 'blue', pch = 16)

  legend('bottomleft', legend = c('pumps', 'deaths'), col = c('blue', 'red'), pch = 1
6)
}
```

First we will calculate the average x, y coordiantes and then plot a circle around this point.

```
mean_x = mean(deaths_points_file$x)
mean_y = mean(deaths_points_file$y)

sd = sqrt(sum((deaths_points_file$x - mean_x)^2 + (deaths_points_file$y - mean_y)^2)/
length(deaths_points_file$x))

plot_base_snow_plot()
bearing = 1:360*pi/180
cx = mean_x + sd * cos(bearing)
cy = mean_y + sd * sin(bearing)
circle <- cbind(cx, cy)
lines(circle, col = 'green', lwd = 2)
```



In the lecture, we calculated the distribution of distances from two sample pumps to all of the cases. Now we will calculate the distances from all other points and identify if the cases really were closer to the Broad Street Pump relative to all other pumps. We will do this by comparing the mean distance.

```
numb_pumps<-nrow(pumps_points_file)
mean_distance<-rep(NA, numb_pumps)
for(jj in 1:numb_pumps){
  single_pump_coord<-pumps_points_file[jj,]
  euc_dist_from_single_pump<-sqrt((deaths_points_file$x - single_pump_coord$x)^2 + (d
eaths_points_file$y - single_pump_coord$y)^2)
  mean_distance[jj] = mean(euc_dist_from_single_pump)
}
```

How does the mean distance from Broad Street pump compares to the other pumps?

- TO DO: Would you make the same inference if you compared the distributions of the distances (not just the means)? How would you compare these?

Vaccination coverage in Madagascar

Now, we will read in data from the 2008-2009 Madagascar Demographic Health Survey (in the file: Madagascar2008-2009.csv). These large scale, cross-sectional surveys are conducted globally, freely available, and may contain relevant covariates. We will first bring the data:

```
library(mgcv)
```

```
## Loading required package: nlme
```

```
##
## Attaching package: 'nlme'
```

```
## The following object is masked from 'package:raster':
##
##     getData
```

```
## This is mgcv 1.8-17. For overview type 'help("mgcv-package")'.
```

```
dhs<-read.csv('Madagascar2008-2009.csv', header = TRUE)
good<-which(dhs$long != 0 & dhs$lat != 0, arr.ind = TRUE)
```

We will focus on the variables ‘age’, ‘measles.y’, and the geographic location (‘long’ and ‘lat’). measles.y is coded as a 1 if the mother can report on whether the child was vaccinated, otherwise it is coded as a 0. Some of the coordinates are incorrect, so let’s make a new variable identifying which rows in the data set have true longitude and latitude coordinates.

```
good<-which(dhs$long != 0 & dhs$lat != 0, arr.ind = TRUE)
```

With this data, we can fit a very simple non-linear statistical model to vaccination coverage, which essentially ‘smooths’ across age, and ‘smooths’ across space. This is clearly very simplistic! But is presented here as a starting point from which further analyses could proceed. Here, we use the package mgcv, which fits ‘generalized additive models’ (or gams, see Wood et al. 2015), following a syntax much like the regression syntax in R, but where ‘s’ indicates ‘smoothed’ terms. Type ?gam into your console if you want to learnmore about this function. We fit this model keeping only the ‘good’ values defined above in the data-set.

```
fit<-gam(measles.y~s(age.in.months)+s(long,lat), family = 'binomial', data = dhs[good
,])
```

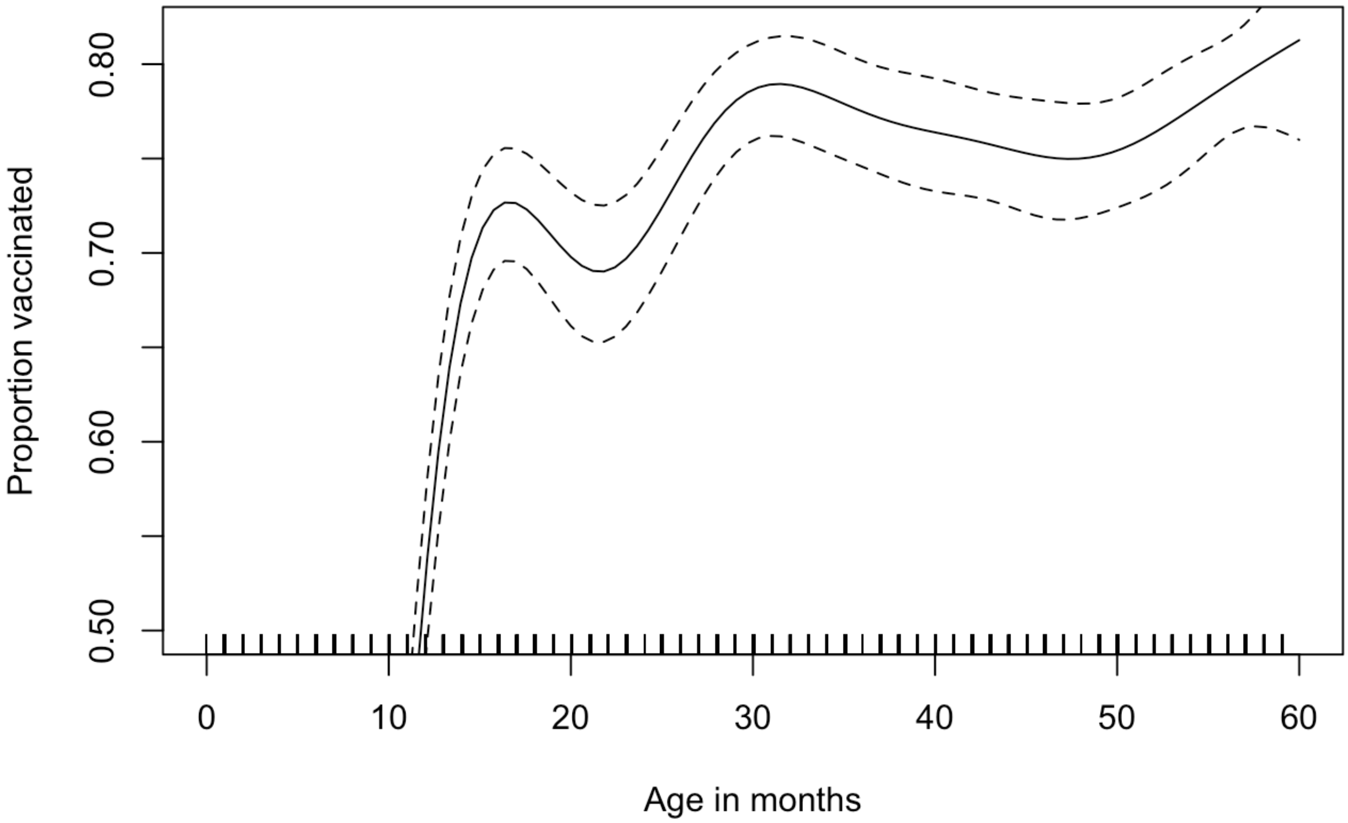
and we can see if these covariates significantly explain the patterns:

```
summary(fit)
```

```
##
## Family: binomial
## Link function: logit
##
## Formula:
## measles.y ~ s(age.in.months) + s(long, lat)
##
## Parametric coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.09343    0.05363   1.742   0.0815 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Approximate significance of smooth terms:
##                     edf Ref.df Chi.sq p-value
## s(age.in.months)  8.389  8.859 1173.6  <2e-16 ***
## s(long,lat)      28.044 28.938  856.2  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## R-sq.(adj) =  0.386   Deviance explained = 33.2%
## UBRE = -0.097253  Scale est. = 1         n = 11466
```

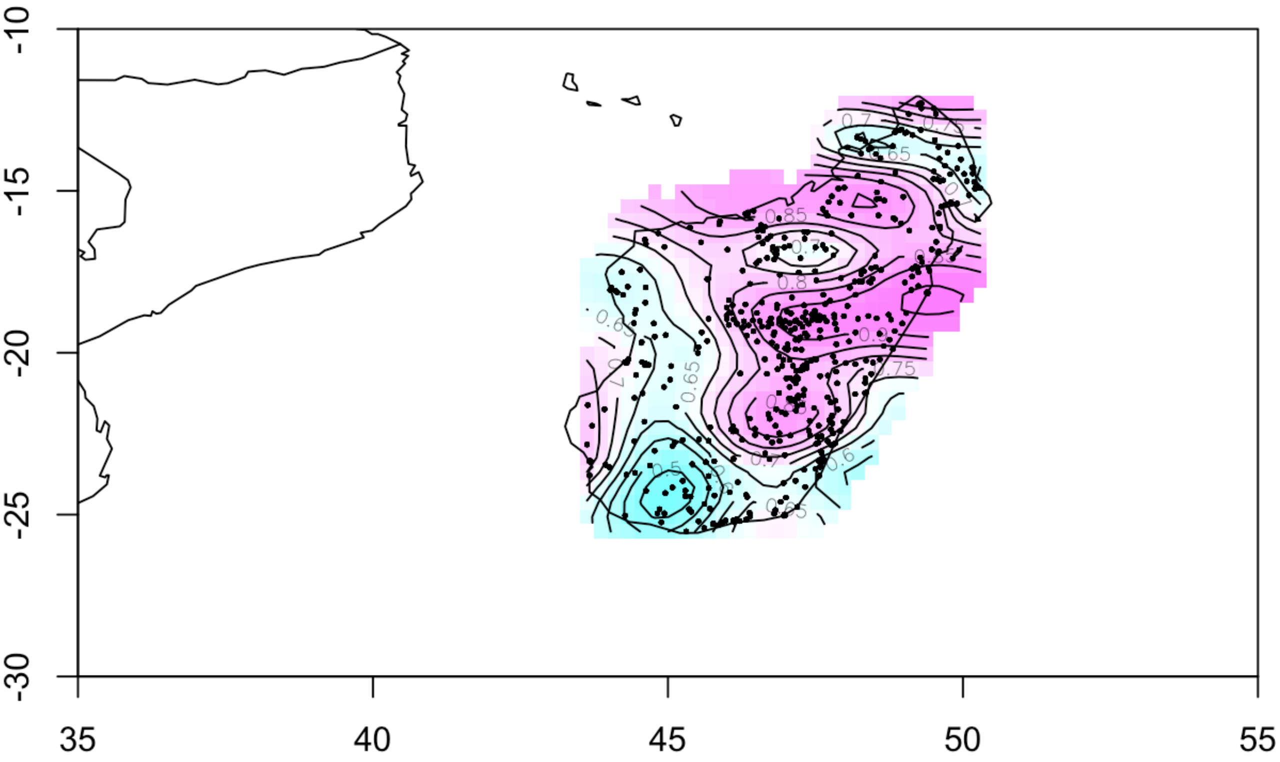
The stars tell us that both of these covariates significantly improve model fit. We can therefore plot the projected patterns over age and space inferred by this model, first looking at the predicted pattern over age:

```
plot(fit, select = 1, trans = function(x)exp(x)/(1+exp(x)), xlim = c(0,60), ylim = c(
0,1.5), xlab = 'Age in months', ylab = 'Proportion vaccinated')
```



Note that since we are using binomial (0,1) data, we've effectively fitted a logistic transform, so the function 'trans' is used to bring our results back to the 0,1 scale. The pattern predicted by the model broadly makes sense - most vaccination is delivered in very young kids (aged < 12 months) and then vaccination rates tail off. We can also plot patterns across space:

```
library(maps)
vis.gam(fit,view=c("long","lat"),plot.type="contour",too.far=0.1,type="response",color="cm",ylim=c(-30,-10),xlim=c(35,55), xlab="", ylab="", main="")
points(dhs$long[good], dhs$lat[good], pch=19,cex=0.2)
map(add=TRUE)
```



This again, broadly, makes sense - in Madagascar, it is reasonable to expect that the highest coverage (shown here in purple) will be around the capital, Antananarivo, which is towards the center of the country; and lower coverage (blue) in the south. To figure out how many susceptible children this distribution of vaccination coverage will result in (which then is of relevance for the SIR models, as we can project incidence through time, but also more formally defines the population at risk, and lets us known where most vulnerable children are to be found) we could bring in layers including the details of population density of children aged <5, or birth rates across space from worldpop.org.uk. See for example Takahashi et al. (2015).

Issues to consider with Generalized Additive Models

We use gams here as a descriptive tool, and do not go into the details of issues associated with over-fitting, out-of-data prediction, choice of smooth terms, etc, but these should all be considered for more serious use of these methods!

- TO DO: Do you see the same patterns with the three polio vaccines? Why would you (or wouldn’t you) expect the same relationships?

Calculating travel times and routes between destinations using a google api

Below are some additional mapping packages. One common question is: how to calculate the distance between two points. We can use the googleway library to calculate the driving distance between locations. These results will be biased based on the quality of the road data and estimated driving times in google, so they may not be very accurate for particular areas of the world. But they will provide a rough estimate of the time it takes to go between locations, which will likely be an improvement on just Euclidean distance. The package googleway allows you to access the google api. Additional information is available here: https://cran.r-project.org/web/packages/googleway/vignettes/googleway-vignette.html (https://cran.r-project.org/web/packages/googleway/vignettes/googleway-vignette.html). We will use multiple libraries to first 1) calculate the travel distance between the airport in Tana and Mahajanga, and then 2) plot the route on a google map.

First, we will load the libraries:

```
library(googleway)
library(leaflet)
library(raster)
```

Next, we will use a google API code (which is also available at: https://developers.google.com/maps/documentation/javascript/ (https://developers.google.com/maps/documentation/javascript/))

```
## Google directions API
key<-"AIzaSyCuhQ4K6pVY1jl62qXvAmhNTiw3GQDoSKk"
```

Now, we will calculate the travel distance between the Antananarivo Airport, Antananarivo, Madagascara nd Mahajanga, Madagascar. This example will only include a single origin and destination, but you can easily adapt the code to calculate the distance between multiple pairs of locations (change the origin to a list of elements, similarly change the destination to a list of elements). You can list locations by their name, address, or longitude/latitude. However, if google has a hard time locating the name of a location, then the code will not work. It will be the most reliable to use longitude/latitude values. You may also change the mode of transport see (help(google_distance)).

```
## get the travel distance
test_distance<- google_distance(origin = "Antananarivo Airport, Madagascar", destinat
ion = "Mahajanga, Madagascar", key = key, mode = "driving")

driving_distance<-test_distance$rows
print(driving_distance)
```

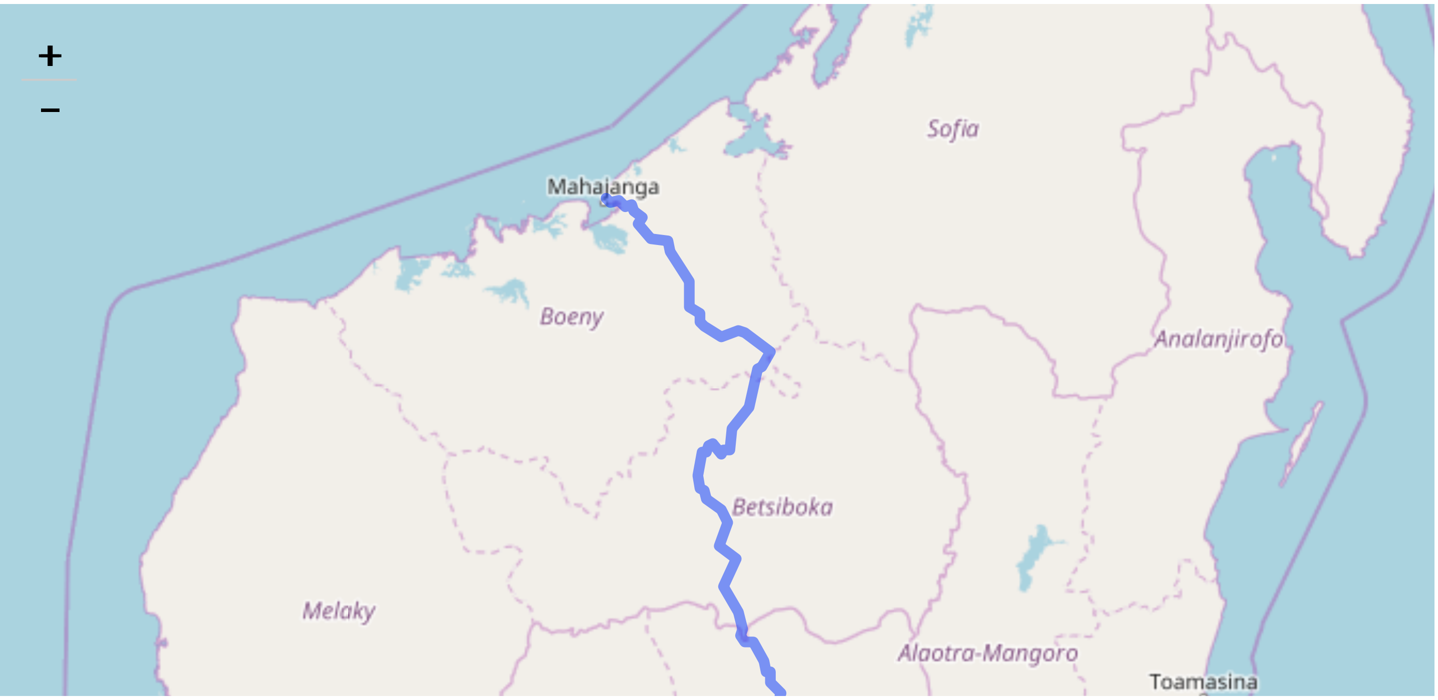
```
##                                                                  elements
## 1 542 km, 542224, 8 hours 28 mins, 30465, 8 hours 28 mins, 30464, OK
```

Next, we will plot the route on a map from open streetmap.

```
test_route<- google_directions(origin = "Antananarivo Airport, Madagascar", destinati
on = "Mahajanga, Madagascar", key = key, mode = "driving")

# get encoded route
route <- test_route$routes$overview_polyline$points
test_route <- decode_pl(route)
head(test_route)
```

```
##          lat      lon
## 1 -18.79970 47.47521
## 2 -18.81748 47.46686
## 3 -18.82780 47.45609
## 4 -18.82094 47.44351
## 5 -18.81709 47.43871
## 6 -18.81017 47.43400
```

```
# plot result
map <- leaflet() %>%
  addTiles("http://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png")

map %>% addPolylines(test_route$lon, test_route$lat)
```



Analamanga
Bongolava
Antananarivo
Itasy
Atsinanana
Madagascar
Vakinankaratra
Leaflet (http://leafletjs.com)