Mapping “Badness”

“Even before you understand them, your brain is drawn to maps”

Ken Jennings, author and Jeopardy champ

We have looked Maps have always been in the realm of cartographers and artists, but the barrier to entry realm has been steadily opening over the last few decades. Technology computers and the Internet , now any yahoo can map out data on Google (that play on words was odd and intentional). And as we saw in Chapter 4, it’s possible to pull geo-location of IP addresses and get spatial data, and that’s what we’re going to focus on.

How do maps help us within information security?

The data we are going to use for the first half of this chapter comes to us from Symantec, who shared a list of clients infected with the ZeroAccess Botnet as collected over a 24 hour period. But they didn’t share the IP But rather than share the IP address

The question we’ll really want to target in this chapter though, is whether or not we should map the virtual world into the physical?

24 hour period

Who is more likely to get a bot infection?

Who is more likely to be infected with ZeroAccess?

Is some proportion of the population more at risk of a ZeroAccess infection?

Simplifying Maps

It’s easy to get all wrapped up thinking that visualizing spatial data (maps) are special, complicated or will somehow take a lot more effort. But with the right tools (and there are plenty available) working with spatial data can not only be relatively simple, but pretty darn fun. In order take some of the mystique out of maps, let’s start by loading up the latitude and longitude points we got from Symantec and just treat them as x,y coordinates and create a simple scatter plot:

# read the CSV with headers

**za <- read.csv("ch5/data/zeroaccess.csv", header=F)**

# create a scatter plot ggplot instance

**gg <- ggplot(data=za, aes(x=long, y=lat))**

# now just add the points, set transparency to 1/40th

**gg <- gg + geom\_point(size=1, color="#000099", alpha=1/40)**

# add axes labels

**gg <- gg + xlab("Longitude") + ylab("Latitude")**

# simplify the theme for aesthetics

**gg <- gg + theme\_bw()**

**print(gg)**

Figure 5.1 Basic Scatterplot using Latitude and Longitude [FILENAME 793725c05f001]

See the map created out of points in figure 5.1? This works with our data because we have over 800,000 data and one point is covering more than a large city. We made it a little less menacing by setting the alpha (transparency of the color as we covered in chapter 3) to be 1/40th of a full color. But just from this basic scatter plot, we can see the density in the eastern half and west coast of the U.S. and most of Europe is covered. We see some concentration in Brazil, and India is outlined quite well. One interesting thing to note here is that China has almost no density and Japan is clearly visible. At this point, we can only make guesses as to what’s going on with that huge looking difference, but we will end up digging a bit deeper here.

Now comes the secret sauce about maps: it’s all about the projection. Everyone is familiar with the Cartesian coordinate system, if not by name then by sight. It’s the fancy name for coordinate system of the simple x,y plotting we just did. By plotting the latitude and longitude points as same-spaced x and y coordinates, we can see the map, but it looks a little odd. South America looks like it is stretched long for example. And this is where map projections come in because we need to “project” a three-dimensional spherical world onto a two-dimensional flat canvas. This creates some problems since there are multiple ways to do that projection as we see in figure 5.2

But creating points on the canvas isn’t all that helpful, we want to have a full canvas to populate. Luckily, within R, map data already exists and we can load up map data with a single command:

# load map data of the world

**world <- map\_data(“world”)**

This loads up just over 25 thousand rows describing various things about the world map (less after we filter out Antartica). We can view countries and borders of the world map by tracing a path on the lat/long pairs in the map data. We have to group the paths by the column labeled group (which in this data groups the country). We will then call coord\_map() to create the map projections and we’ll use a simple black and white theme on it. To make this map useful, we’ll also plot the points from the ZeroAccess data on the map.

# load map data of the world

**world <- map\_data("world")**

# nothing personal penguins, but strip out Antarctica

**world <- subset(world, world$region!="Antarctica")**

# load world data into ggplot object

**gg <- ggplot(data=world, aes(x=long, y=lat))**

# trace along the lat/long coords by group

**gg <- gg + geom\_path(aes(group=group), colour="gray70")**

# now project using the mercator projection

**gg <- gg + coord\_map("mercator", xlim=c(-200, 200))**

# load up the points by overiding the default data set

**gg <- gg + geom\_point(data=za, aes(long, lat),**

**colour="#000099", alpha=1/40, size=1)**

# add axes labels and theme

**gg <- gg + xlab("Longitude") + ylab("Latitude")**

**gg <- gg + theme\_bw()**

**print(gg)**

Figure 5.3 Worldwide ZeroAccess Infections [FILENAME 793725c05f003]

Now that’s a real map, but what can we learn from it? The answer is not much, it doesn’t tell us much more than the ZeroAccess botnet is an international traveller (and who cares). So let’s try something different. Let’s attempt to categorize by country and we can visualize that with a chloropleth. We’ll have to first figure out which country the points are in though. And for that we’ll use a function in the sudabook package called latlong2map(). That function will take in a data frame of longitude and latitude pairs and the name of a map to translate onto. We will get back a vector of country names in this case and then we want to count up how many times the country appears with the table() command. Next we’ll want to merge() the count of countries with the map data (then reorder it), this allows us generate a map and connect the fill aesthetic to the count of bot infections in each country.

# convert long/lat into country names

**zworld <- latlong2map(data.frame(x=za$long, y=za$lat), "world")**

# count up points in the country and conver to data frame

**wct <- data.frame(table(zworld))**

# label the country as "region" to match map data

**colnames(wct) <- c("region", "count")**

# merge will match on "region" and add "count"

**za.chloro <- merge(world, wct)**

# now we sort, otherwise it's disasterous

**za.chloro <- za.chloro[with(za.chloro, order(group, order)), ]**

# and plot

**gg <- ggplot(za.chloro, aes(x=long, y=lat, group=group, fill=count))**

**gg <- gg + geom\_path(colour="#666666") + geom\_polygon()**

**gg <- gg + coord\_map("mercator", xlim=c(-200, 200), ylim=c(-60,200))**

**gg <- gg + scale\_fill\_gradient2(low="#FFFFFF", high="#4086AA",**

**midpoint=median(za.chloro$count))**

**gg <- gg + theme\_plain()**

**print(gg)**

Figure 5.4 Chloropleth of ZeroAccess Infections [FILENAME 793725c05f004]

And voila, we get a rather good-looking map that makes it look like the U.S. has the market cornered on ZeroAccess infections. But because it’s very difficult to tell quantity by color density (we’ll cover that in Chapter 6), we want to take a step back for a moment and get a feel for the proportion of infections in the U.S. Look back at the wct variable we created above. Let’s see what that data looks like:

**head(wct)**

region count

1 Afghanistan 53

2 Albania 1166

3 Algeria 3014

4 Andorra 4

5 Angola 160

6 Argentina 6016

We can use the wct variable to get a feel for the proportion of infections in the U.S.:

# for each wct$count, divide by sum

**perc <- wct$count/sum(wct$count)**

# covert to a readable format

**wct$perc <- round(perc, 4)\*100**

# now order the highest proportions on top

**wct <- wct[with(wct, order(perc, decreasing=T)), ]**

# look at the top few entries.

**head(wct)**

# output:

region count perc

148 USA 261627 35.23

24 Canada 35607 4.79

74 Japan 33590 4.52

145 UK 31813 4.28

50 Germany 27336 3.68

71 Italy 25717 3.46

Wow, the U.S. has 35% of the ZeroAccess infections and the next most represented country is Canada at just under 5%. No wonder our chloropleth was so slanted towards the U.S.

In order to get to the heart of our question, “Who is more likely to be infected with ZeroAccess?” We want to simplify our data for the U.S. and understand how that may change our question. We want to do this not just because working with over 800,000 data points can be a bit slow on some systems, but also it will be much easier for us to focus in on the U.S. because of our knowledge of the geography and accessibility of data for us. So let’s focus on the U.S. If we attempt to plot a U.S. map and then project all the points on it, we get a rather funny picture because it will still plot all of the world points in the data set. Therefore we have to remove non-U.S. states. We can use the latlong2map() command again and this time, map the points to U.S. states and anything that does not get mapped to a U.S. state will be assigned as NA, which can then be filtered.

**zstate <- latlong2map(data.frame(x=za$long, y=za$lat), "state")**

# select rows from za where the zstate is not NA

**za.state <- za[which(!is.na(zstate)), ]**

And now we can make a pretty map of the continental U.S. showing the ZeroAccess bot infections. But before we do that, let’s create a theme for ggplot2 that will not show the axes or box around it, and have just a plain map. By creating this, we can just add on theme\_plain() instead of the built-in theme\_bw() when we call ggplot().

# create a plain theme for ggplot maps

**theme\_plain <- function() {**

**theme(axis.title = element\_blank(),**

**axis.text = element\_blank(),**

**panel.background = element\_blank(),**

**panel.grid = element\_blank(),**

**axis.ticks.length = unit(0, "cm"),**

**axis.ticks.margin = unit(0, "cm"),**

**panel.margin = unit(0, "lines"),**

**plot.margin = unit(c(0,0,0,0), "lines"),**

**complete=TRUE)**

**}**

# load map data of the U.S.

**state <- map\_data("state")**

**gg <- ggplot(data=state, aes(x=long, y=lat))**

**gg <- gg + geom\_path(aes(group=group), colour="gray80")**

**gg <- gg + coord\_map("mercator")**

**gg <- gg + geom\_point(data=za.state, aes(long, lat),**

**colour="#000099", alpha=1/40, size=1)**

**gg <- gg + theme\_plain()**

**print(gg)**

Figure 5.5 ZeroAccess Infections in the U.S. [FILENAME 793725c05f005]

Look at Figure 5.5 and think how that looks. Does it look strange? This is where we really have to be careful because after working with spatial data, we can tell you this sure looks like map showing population density and not bot infections. Looking at figure 5.5 we find ourselves asking a slightly different question. **Could ZeroAccess infections just be a reflection of the population?** We could apply a statistical technique called regression analysis (and we will later) but let’s stick with pictures and create another chloropleth, but this time we will break up the data based on the U.S. states

The Potwin effect

But at this point, we are digging deeper than country and we will have to account for something we call “the Potwin effect” after the town by that name in Kansas with a population of 449. The population is important because if we take a look at this data, we’d see we have 12,643 reported ZeroAccess infections in the town of Potwin, Kansas. See, we’ve played this game before and when we first stumbled into this, we spent days trying to understand why Potwin, KS was so odd. We realized that these couldn’t be valid entries and looking at the data, the entries were oddly rounded off to integers and they were all “38,-97”. Finally it dawned on us. IP geo-location services should always know what country an IP address is in because the IANA records are clear about that. But if the geo-location service cannot get any more specifics beyond the country, they return a rounded-off location near the geographic center of the country. In the U.S., the geographic center is just outside of Potwin, KS. For this purpose, they are “unknown U.S. locations” and not really in Kansas, so we are going to remove these data points the next bit of code.

In the next map, we want to use color again to show quantity, but rather than just using a single hue (a fancy term for the color of the rainbow), let’s use a diverging color scheme and assign the mid-point of the range of hues at the mean count per state. This will allow us to show states with above average infection counts with one hue and the below average state with another. As a side note, let’s also change the projection from the Mercator projection to the Polyconic. That projection looks odd at the world level, but puts a nice slope and curve in a U.S. map.

# create a chloropleth of the U.S. states

# because all of these vectors are from the same source (za),

# we can cross the indexes of the vectors

# pull out those that are not NA, and not rounded off in Potwin

**state.index <- which(!is.na(zstate) & za$lat!=38 & za$long!=-97)**

# now create a count of states and filter on those indexes

**sct <- data.frame(table(zstate[state.index]))**

**colnames(sct) <- c("region", "count")**

# merge with state map data

**za.sct <- merge(state, sct)**

# Now plot a choropleth using a diverging color

**colors <- suda.pal(5, "div")**

**gg <- ggplot(za.sct, aes(x=long, y=lat, group=group, fill=count))**

**gg <- gg + geom\_polygon(colour="black")**

**gg <- gg + coord\_map("polyconic")**

**gg <- gg + scale\_fill\_gradient2(low=colors[5], mid=colors[3],**

**high=colors[1],**

**midpoint=mean(za.sct$count))**

**gg <- gg + theme\_plain()**

**print(gg)**

Figure 5.6 Chloropleth of U.S. States with ZeroAccess [FILENAME 793725c05f006]

Wow, this is looking nice, but don’t get too excited as this is relatively useless for comparing the states. We can easily see that California, Texas, Florida and New York are above average, but it’s also worthwhile to know that the 4 most populated states are also California, Texas, New York and Florida in order. In other words, we may just be seeing a reflection of population in this map, so we have to normalize this data to the population. In order to normalize we could take a multiple approaches. The two simplest approaches are to normalize the population to the infections and ask “how many people per bot infection?” or we could reverse that and ask “how many bot infections per person”. Well in this case, the latter question would give us a small fraction since we have far less infections then people, so we’ll do the first method to work with integers.

We will want to eventually merge the bot infection data with other data to answer our question.