SOUTHERN ILLINOIS UNIVERSITY EDWARDSVILLE

CMIS 427 Security Analytics

Assignment 7

Performing Exploratory Security Data Analysis with R scripts

Author:
Luis Barrios
MS CMIS Student
Professor:
Dr. Joseph Vithayathil

April 9, 2022

Table of Contents

Performing Exploratory Security Data Analysis with R scripts

This report consists in creating an analysis using maps as main visualization tool to mining the data. The analysis uses 3 data sets containing geo coordinates, population attributes, and infection points.

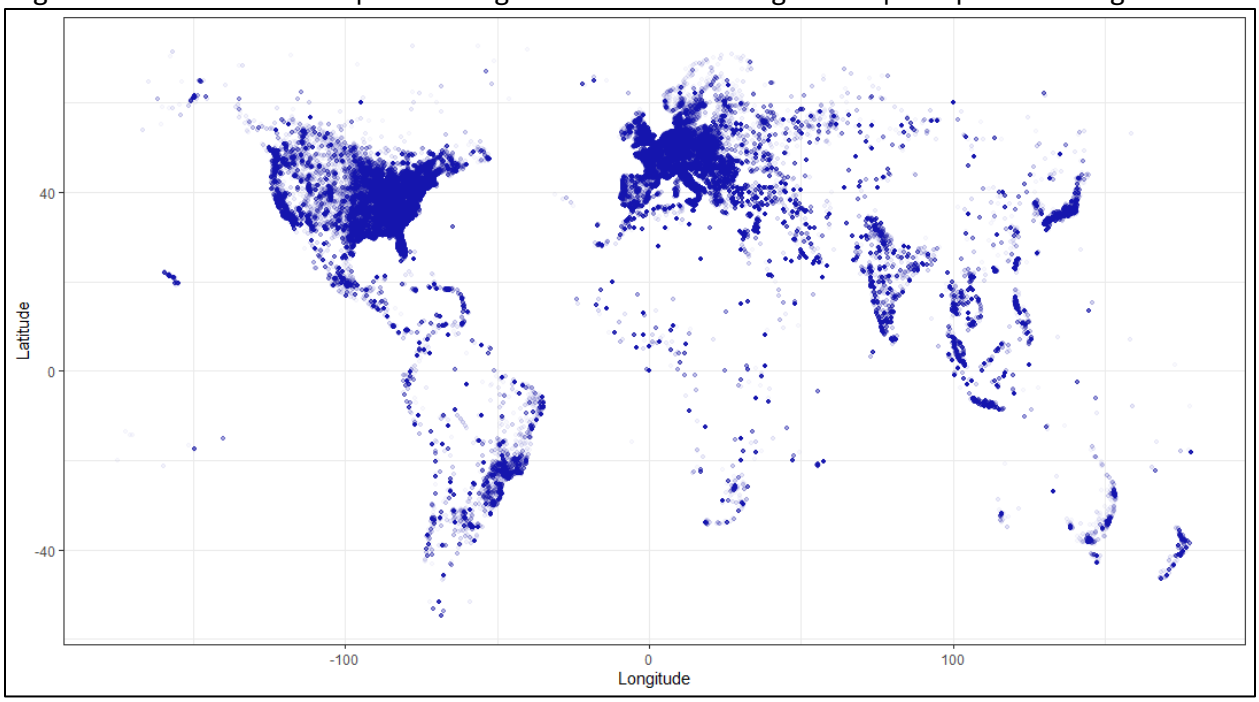
Listing 5-0: consist in uploading all the required libraries to set the Rstudio environment

```
# Listing 5-0 ##############################################################
# requires packages:
library(ggplot2)
library(maps)
library(RColorBrewer)
library(scales)
library(reshape2)
library(grid)
library(sp)
library(maptools)
library(car)
```

Listing 5-1: Consist in reading the zeroaccess.csv file which contains geo coordinates and plot the point on a map. See figure 1 for output reference.

```
# Listing 5-1 ##############################################################

# read the CSV with headers
za <- read.csv("C:/Users/luisa/Documents/SIUE/Courses/CMIS
427/code/9781118793725_download/book/ch05/data/zeroaccess.csv", header=T)

# create a ggplot instance with zeroaccess data
gg <- ggplot(data=za, aes(x=long, y=lat))
# 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()
# this may take a while, over 800,000 points plotted
print(gg)
```

Figure 1: Basic scatterplot using latitude and longitude | Output listing 5-1

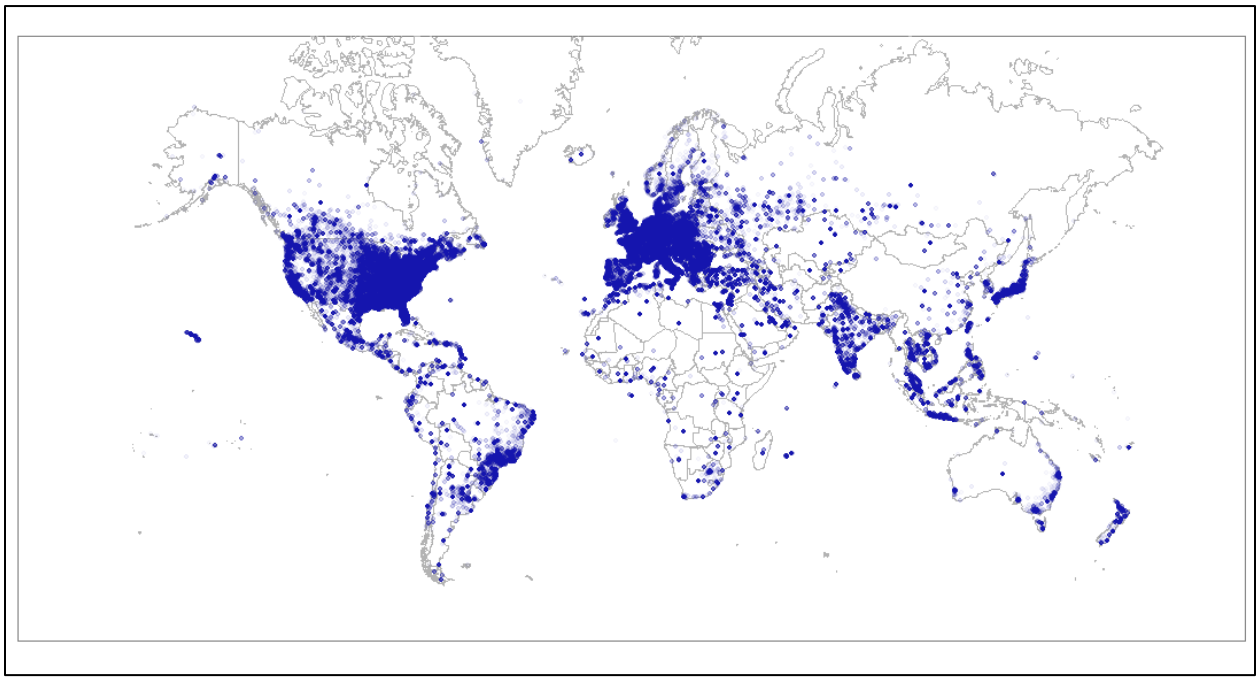


Listing 5-2: if you detail figure 1, you will notice that map is composed only with the data points. This listing uses ggplot2 to plot the coordinates in map chart with country separation lines. See figure 2 for output reference

```
# Listing 5-2 ##############################################################
# requires package : ggplot2
# requires object: za (5-1)
# the "maps" and "mapproj" packages are used by ggplot
# 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 (countries)
gg <- gg + geom_path(aes(group=group), colour="gray70")
# now project using the mercator projection
# try different projections with ?mapproject
gg <- gg + coord_map("mercator", xlim=c(-200, 200))
# load up the ZeroAccess points, overiding the default data set
gg <- gg + geom_point(data=za, aes(long, lat),
                      colour="#000099", alpha=1/40, size=1)
# remove text, axes ticks, grid lines and do gray border on white
```

```
gg <- gg + theme(text=element_blank(),
                 axis.ticks=element_blank(),
                 panel.grid=element_blank(),
                 panel.background=element_rect(color="gray50",
                                               fill="white"))
print(gg)
```

Figure 2: Scatterplot with ggplot 2 and coordinates | output for listing 5-2



Listing 5-3: consist in creating a choropleth for the previous data in order to make the graph more legible. However, this listing only set the functions required to graph. You can find the function at Ryan Weald's github.

```
# Listing 5-3 ##############################################################
# slightly modified verison of Ryan Weald's (@rweald) function
# https://gist.github.com/rweald/4720788
latlong2map <- function(pointsDF, mapping) {
  # load up the map data
  local.map <- map(mapping, fill=TRUE, col="transparent", plot=FALSE)
  # pull out the IDs from the name
  IDs <- sapply(strsplit(local.map$names, ":"), function(x) x[1])
  # Prepare SpatialPolygons object
  maps_sp <- map2SpatialPolygons(local.map, IDs=IDs,
                                 proj4string=CRS("+proj=longlat
+datum=wgs84"))
  # Convert pointsDF to a SpatialPoints object
```

```
  pointsSP <- SpatialPoints(pointsDF,
                            proj4string=CRS("+proj=longlat +datum=wgs84"))
  # Use 'over' to get _indices_ of the Polygons object containing each point
  indices <- over(pointsSP, maps_sp)
  # Return the names of the Polygons object containing each point
  mapNames <- sapply(maps_sp@polygons, function(x) x@ID)
  # now return a vector of names that match the points
  mapNames[indices]
}
```

Listing 5-4: consist in creating a choropleth for the previous data in order to make the graph more legible. It requires listing 5-3; see figure 3 for output reference.

```
# Listing 5-4 ##############################################################

# requires objects: za (5-1), world (5-2), latlong2map (5-3)
# convert ZeroAccess long/lat into country names from world map
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" in each and add "count" to "world"
za.choro <- merge(world, wct)
# now we sort the map data to original sequence
# otherwise the map is disasterous
za.choro <- za.choro[with(za.choro, order(group, order)), ]
# and plot
gg <- ggplot(za.choro, 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.choro$count),
                                name="Infections")
# remove text, axes ticks, grid lines and do gray border on white
gg <- gg + theme(axis.title=element_blank(),
                 axis.text=element_blank(),
                 axis.ticks=element_blank(),
                 panel.grid=element_blank(),
                 panel.background=element_rect(color="gray50",
                                               fill="white"))
print(gg)
```

Figure 3: Choropleth of Zero Access infections | Output for listing 5-4

Listing 5-5: this script counts the number of points per country and gives a proportion as a whole to understand how the ZeroAccess data is distributed in percentages. See figure 4 for output reference.

```
# Listing 5-5 ##############################################################
# requires object: wct (5-4)
head(wct)

# for each wct$count, divide by sum, gives us proportion of the whole
perc <- wct$count/sum(wct$count)
# covert to a readable format, round it and create percent
wct$perc <- round(perc, 4)*100
# now order the highest percentages on top
wct <- wct[with(wct, order(perc, decreasing=T)), ]
# look at the top few entries.
head(wct)
```

Figure 4: Output for listing 5-5

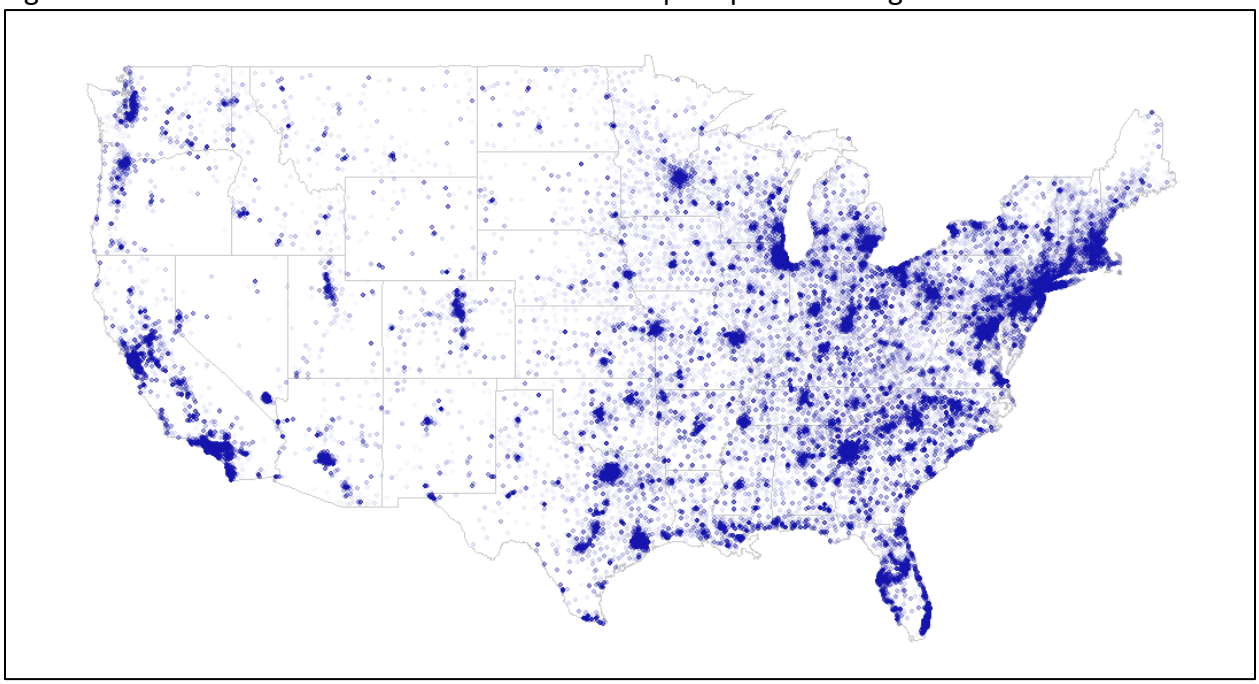
```
> # Listing 5-5 ##############################################################
> # requires object: wct (5-4)
> head(wct)
       region count
1 Afghanistan    53
2     Albania  1166
3     Algeria  3478
4     Andorra     6
5      Angola   160
6    Anguilla     5
>
>
> # for each wct$count, divide by sum, gives us proportion of the whole
> perc <- wct$count/sum(wct$count)
> # covert to a readable format, round it and create percent
> wct$perc <- round(perc, 4)*100
> # now order the highest percentages on top
> wct <- wct[with(wct, order(perc, decreasing=T)), ]
> # look at the top few entries.
> head(wct)
     region  count  perc
205     USA 275333 35.43
35   Canada  35904  4.62
93    Japan  34313  4.42
201      UK  32270  4.15
90    Italy  29344  3.78
68  Germany  27442  3.53
>
```

Listing 5-6: consist in filtering down our map to The United States. In this way we can take a better look of the map. See figure 5 for output reference.

```
# Listing 5-6 ##############################################################

# requires objects: za (5-1), latlong2map (5-3)
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)), ]
# 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)
# stripping off the "chart junk"
gg <- gg + theme(axis.title=element_blank(),
                 axis.text=element_blank(),
```

```
                    axis.ticks=element_blank(),
                    panel.grid=element_blank(),
                    panel.background=element_blank())
print(gg)
```

Figure 5: Zeroaccess Infections in the United State | Output for listing 5-6

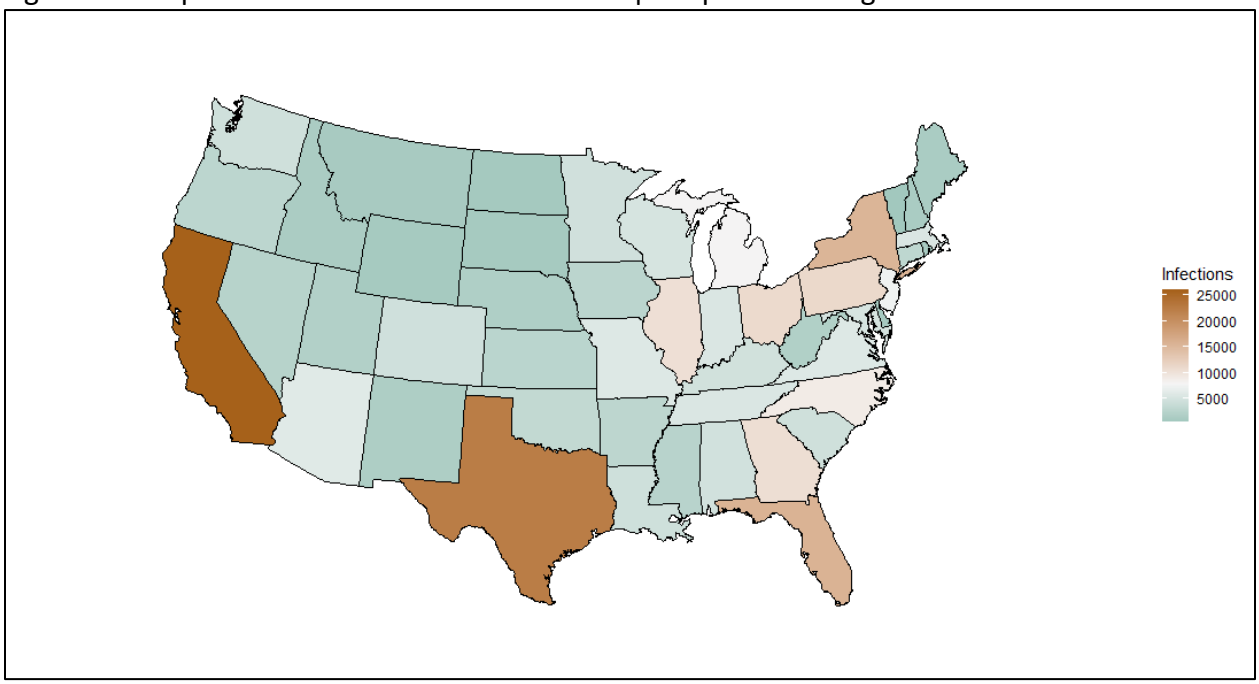


Listing 5-7: consist in upgrading the previous figure by adding a diverging color scheme in the choropleth to differentiate infections from state to state. See figure 6 for output reference.

```
# Listing 5-7 ##############################################################
# requires objects: za (5-1), latlong2map (5-3)

# create a choropleth of the U.S. states
# because all of these vectors are from the same source (za),
# we can cross the indexes of the vectors
zstate <- latlong2map(data.frame(x=za$long, y=za$lat), "state")
# pull out those that are not NA, and take care of Potwin effect
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
```

```
# from the dds package
# cat(dds.pal(5, "div"))
colors <- c("#A6611A", "#DFC27D", "#F5F5F5", "#80CDC1", "#018571")
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),
                                name="Infections")
gg <- gg + theme(axis.title=element_blank(),
                 axis.text=element_blank(),
                 axis.ticks=element_blank(),
                 panel.grid=element_blank(),
                 panel.background=element_blank())
print(gg)
```

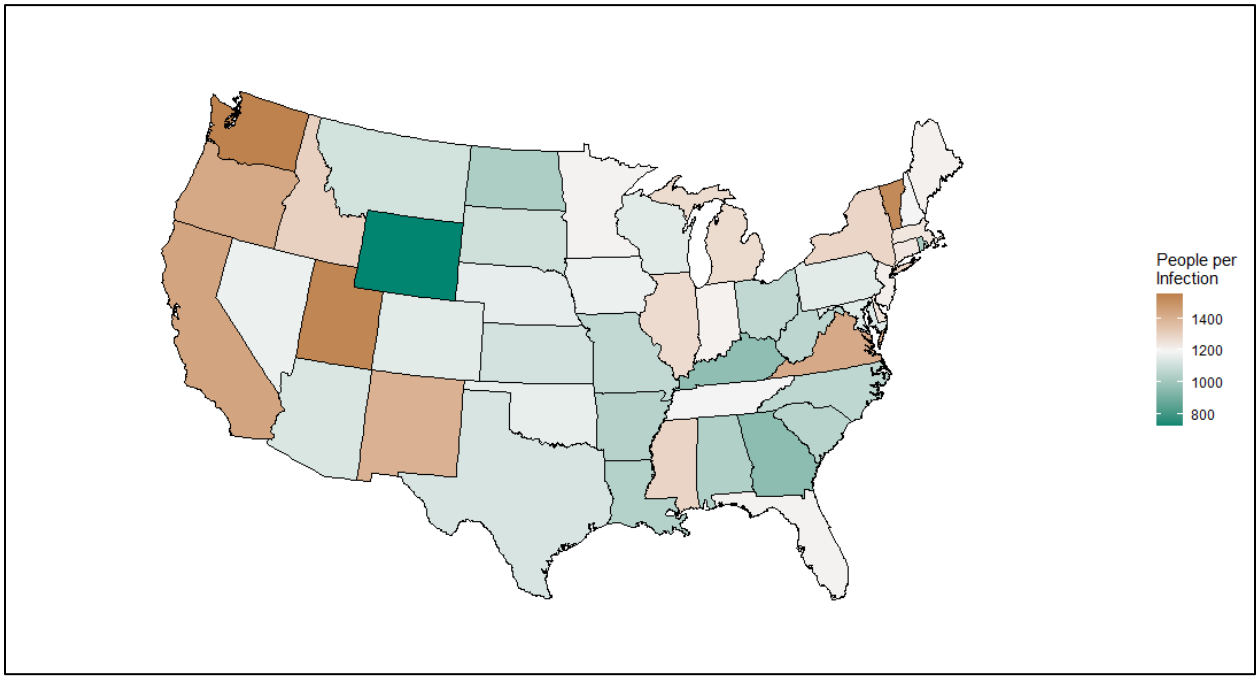
Figure 6: chorpleth of U.S sates with ZeroAccess | Output for listing 5-7



Listing 5-8: consist in normalizing the data from listing 5-7 because it does assign percentage of population which means the map would flag states with most population. In this case, we add number of users per state and merge it with the previous data to get a more meaning full map. See figure 7 for output reference.

```
# Listing 5-8 ##############################################################
# requires objects: sct (5-7), colors (5-7), latlong2map (5-3)
# read in state population and internet users
# data scraped from http://www.internetworldstats.com/stats26.htm
users <- read.csv("C:/Users/luisa/Documents/SIUE/Courses/CMIS
427/code/9781118793725_download/book/ch05/data/state-internets.csv", header=T)

# all the state names are lower case in map data, so convert
users$state <- tolower(users$state)

# now merge with the sct data from previous example
# merge by sct$region and users$state
za.users <- merge(sct, users, by.x="region", by.y="state")

# calculate people to infection
# change this to internet users if you would like to try that
za.users$pop2inf <- round(za.users$population/za.users$count, 0)

# and create a simple data frame and merge
za.norm <- data.frame(region=za.users$region,
                      count=za.users$pop2inf)
za.norm.map <- merge(state, za.norm)

# now create the choropleth
gg <- ggplot(za.norm.map, 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.norm.map$count),
                                name="People per\nInfection")
gg <- gg + theme(axis.title=element_blank(),
                 axis.text=element_blank(),
                 axis.ticks=element_blank(),
                 panel.grid=element_blank(),
                 panel.background=element_blank())
print(gg)
```

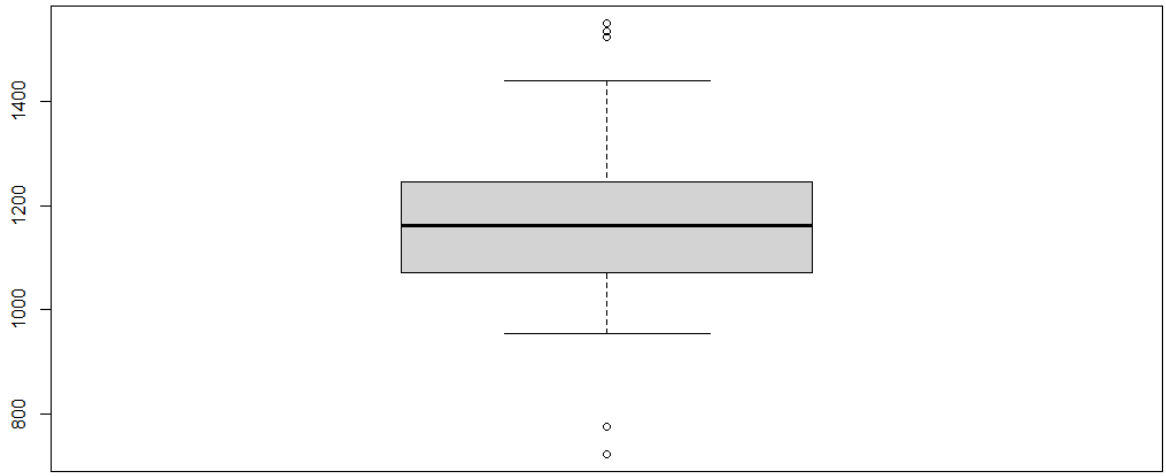
Figure 7: Normalized ZeroAccess infections: Number of people in the state per one infection | Output for listing 5-8



Listing 5-9: consist in creating a Boxplot to find outliers in the dataset. The further the points are, the more likely to be outliers. See figure 8 for output reference.

```
# Listing 5-9 ##############################################################
# requires objects: za.norm (5-8)

# create a box plot of the count
popbox <- boxplot(za.norm$count)
```

Figure 8: Distribution of normalized State Infections



Listing 5-10: In this case, the script prints out information from the outliers mentioned in the previous listing. See figure 10 for output reference.

```
# Listing 5-10 ##########################################################
# requires objects: za.norm (5-8), popbox (5-9)
# the values that are considered outliers
print(popbox$out)


# pull the rows from za.norm that have those values
za.norm[za.norm$count %in% popbox$out, ]
```

Figure 10: output for listing 5-10

```
> # Listing 5-10 ##########################################################
> # requires objects: za.norm (5-8), popbox (5-9)
> # the values that are considered outliers
> print(popbox$out)
[1]  777 1536 1525 1550  724
>
> # pull the rows from za.norm that have those values
> za.norm[za.norm$count %in% popbox$out, ]
                region count    z
8  district of columbia   777 -2.4
43                 utah  1536  2.2
44              vermont  1525  2.1
46           washington  1550  2.2
49              wyoming   724 -2.7
>
>
```

Listing 5-11: consist in identifying the outliers using Zcores. For output reference see figure 10.

```
# Listing 5-11 ##########################################################
# requires objects: za.norm (5-8)

# get the standard deviation
za.sd <- sd(za.norm$count)
# get the mean
za.mean <- mean(za.norm$count)
# now calculate the z-score and round to 1 decimal
za.norm$z <- round((za.norm$count-za.mean)/za.sd, 1)
# we can inspect the "z" variable for the specific z-scores
# pull out values where absolute value of z-score is > 2
za.norm[which(abs(za.norm$z)>2), ]
```

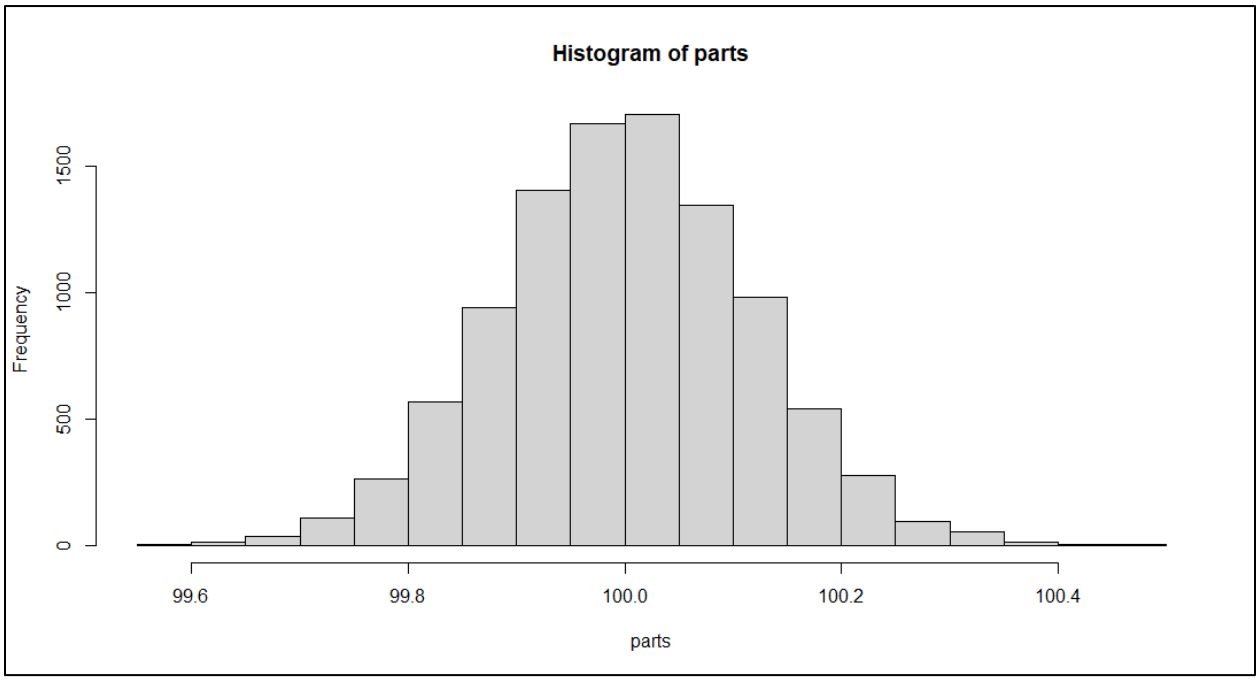
Listing 5-12 and 5-13: Consist in providing a example concerning statistical validation. This is due to the natural variations will cancel out more often than stack up. See figure 11 for output reference. Additionally, we can see the normal distribution of part with a histogram, see figure 12 four output reference.

```
# Listing 5-12 ##############################################################
#setting seed for reproducibility
set.seed(1492)
# run 100 times, getting random values between 98 and 102
mean(runif(100, min=98, max=102))


# Listing 5-13 ##############################################################
#setting seed for reproducibility
set.seed(1492)
# iterate seq(10000) times, generate a set of 100 parts and calc mean
parts <- sapply(seq(10000), function(x) mean(runif(100, min=98, max=102)))
# result is a vector of 10,000 sets
# show the min and max of these parts
range(parts)

# Additional code for histogram
hist(parts)
```

Figure 11: Output for listing 5-12 and 5-13

```
> # Listing 5-12 ##############################################################
> #setting seed for reproducibility
> set.seed(1492)
> # run 100 times, getting random values between 98 and 102
> mean(runif(100, min=98, max=102))
[1] 100.0141
>
>
> # Listing 5-13 ##############################################################
> #setting seed for reproducibility
> set.seed(1492)
> # iterate seq(10000) times, generate a set of 100 parts and calc mean
> parts <- sapply(seq(10000), function(x) mean(runif(100, min=98, max=10
2)))
> # result is a vector of 10,000 sets
> # show the min and max of these parts
> range(parts)
[1]  99.57977 100.47559
>
```

Figure 12: Histogram for part example in listing 5-13 | Additional chart



Listing 5-14: consist in granulate the analysis to the county level. In this case the Script splits the current string and transformed to vectors to create a new matrix. See figure 13 for output reference.

```
# Listing 5-14 ##########################################################
# requires objects: za (5-1), latlong2map (5-3)

## now mapping lat/long down to county
county <- latlong2map(data.frame(x=za$long, y=za$lat), "county")
za.county <- county[which(!is.na(county) & za$lat!=38 & za$long!=-97)]
# count the occurances
county.count <- table(za.county)
# need to convert "county, state" into a data frame
# so we split it out by comma
temp.list <- strsplit(names(county.count), ",")
# convert the list into a vector
temp.list <- unlist(temp.list)
# force the vector into a 2 column matrix, filling row by row
temp.matrix <- matrix(temp.list, ncol=2, byrow=T)
# and now create the data frame with the count of county infections
za.county <- data.frame(temp.matrix, as.vector(county.count))
# finally assign names to the fields
# names match the field names in the county map_data
colnames(za.county) <- c("region", "subregion", "infections")
head(za.county)
```

Figure 13: Output for listing 5-14 || County matrix

```
> # names match the field names in the county map_data
> colnames(za.county) <- c("region", "subregion", "infections")
> head(za.county)
   region subregion infections
1 alabama   autauga         44
2 alabama   baldwin        184
3 alabama   barbour         13
4 alabama      bibb         13
5 alabama    blount         26
6 alabama   bullock         11
>
```

Listing 5-15: consist in adding more data to explore the relationship of the malware infections and additional data points. See figure 14 for output reference.

```
# Listing 5-15 ##########################################################

# requires objects: za.county (5-14)

# read up census data per county
county.data <- read.csv("C:/Users/luisa/Documents/SIUE/Courses/CMIS
427/code/9781118793725_download/book/ch05/data/county-data.csv", header=T)

# notice the all.x option here
za.county <- merge(x=county.data, y=za.county, all.x=T)

# replace all NA's with 0
za.county$infections[is.na(za.county$infections)] <- 0
summary(za.county)
```

Figure 14: statistic summary for listing 5-15

```
> summary(za.county)
  subregion            region               pop              income          ipaddr
 Length:3072        Length:3072        Min.    :     71   Min.   : 19344   Min.   :        0
 Class :character   Class :character   1st Qu.:  11215   1st Qu.: 37793   1st Qu.:     5367
 Mode  :character   Mode  :character   Median :  26047   Median : 43333   Median :    15289
                                       Mean   : 101009   Mean   : 45075   Mean   :   387973
                                       3rd Qu.:  67921   3rd Qu.: 50010   3rd Qu.:    62594
                                       Max.   :9962789   Max.   :120096   Max.   :223441040
    ufo2010          infections
 Min.   :  0.000   Min.   :   0.00
 1st Qu.:  0.000   1st Qu.:   6.00
 Median :  2.000   Median :  17.00
 Mean   :  7.943   Mean   :  83.79
 3rd Qu.:  6.000   3rd Qu.:  56.00
 Max.   :815.000   Max.   :7692.00
>
```

Listing 5-16: consist in creating the data that will be used for linear regression graph in the next listing. See figure 15 for output reference

```
# Listing 5-16 ##############################################################
# for reproducability
set.seed(1)
# generate 200 random numbers around 10
input <- rnorm(200, mean=10)
summary(input)
```

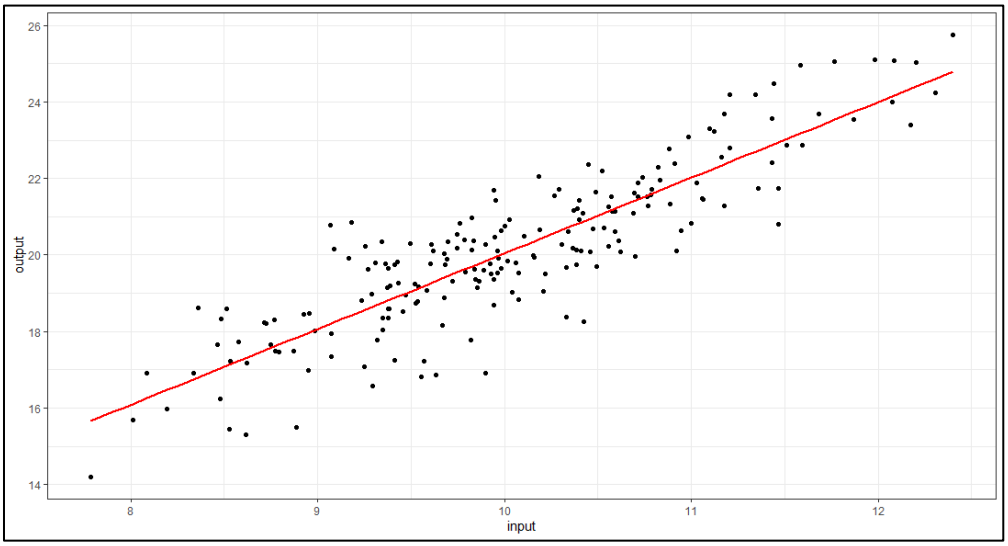
Figure 15: Output for listing 5-16

```
> summary(input)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  7.785   9.386   9.951  10.036  10.613  12.402
>
```

Listing 5-17: consist in creating the plot for the linear regression data and see if there is a relationship. See figure 16 for output reference.

```
# Listing 5-17 ##############################################################
# requires objects: input (5-16)
# generate output around a mean of 2 x input
output <- rnorm(200, mean=input*2)
# put into data frame to plot it
our.data <- data.frame(input, output)
gg <- ggplot(our.data, aes(input, output))
gg <- gg + geom_point()
gg <- gg + geom_smooth(method = "lm", se=F, color="red")
gg <- gg + theme_bw()
print(gg)
```

Figure 16: Sample data with regression line



Listing 5-18: consist in running the linear regression model. See figure 17 four output reference

```
# Listing 5-18 ##########################################################
# requires objects: input (5-16), output (5-17)
model <- lm(output ~ input)
summary(model)
```

Figure 17: Linear regression model for sample data

```
> # Listing 5-18 ##########################################################
> # requires objects: input (5-16), output (5-17)
> model <- lm(output ~ input)
> summary(model)

Call:
lm(formula = output ~ input)

Residuals:
     Min       1Q   Median       3Q      Max
-2.93275 -0.54273 -0.02523  0.66833  2.58615

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.27224    0.77896   0.349    0.727
input        1.97692    0.07729  25.577   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.013 on 198 degrees of freedom
Multiple R-squared:  0.7677,    Adjusted R-squared:  0.7665
F-statistic: 654.2 on 1 and 198 DF,  p-value: < 2.2e-16
```

Listing 5-19: consist in finding the coefficients that represent the standard error. See figure 18 for output reference

```
# Listing 5-19 ##########################################################
# requires objects: model (5-18)
confint(model)
```

Figure 18: coefficients and confident interval for sample data

```
> # Listing 5-19 ##########################################################
> # requires objects: model (5-18)
> confint(model)
                2.5 %   97.5 %
(Intercept) -1.263895 1.808368
input        1.824502 2.129343
```

Listing 5-20: consist in running a linear regression model with real data. In this case, we would use the za.county data fra me that we had created before. Se figure 19 for output reference.

```
# Listing 5-20 ##########################################################
# requires objects: za.county (5-14 and 5-15)
summary(lm(infections ~ ufo2010, data=za.county))
```

Figure 19: Linear regression model on ZeroAccess | Output for listing 5-20

```
> # Listing 5-20 ##########################################################
> # requires objects: za.county (5-14 and 5-15)
> summary(lm(infections ~ ufo2010, data=za.county))

Call:
lm(formula = infections ~ ufo2010, data = za.county)

Residuals:
     Min       1Q   Median       3Q      Max
-1699.73   -25.07   -14.75    -0.75  2738.94

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 17.75484    2.59829   6.833 9.97e-12 ***
ufo2010      8.31413    0.08711  95.445  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 138.8 on 3070 degrees of freedom
Multiple R-squared:  0.7479,    Adjusted R-squared:  0.7479
F-statistic:  9110 on 1 and 3070 DF,  p-value: < 2.2e-16
```

Listing 5-21: The script adds more variable to the previous model in order to explore the data. In this case, the listing adds income, UFOs, and Ip addresses. See figure 20 for output reference.

```
# Listing 5-21 ##########################################################

# requires objects: za.county (5-14 and 5-15)
summary(lm(infections ~ pop + income + ipaddr + ufo2010,
           data=za.county))
```

Figure 20: Linear regression model on ZeroAccess | Output for listing 5-21

```
> # Listing 5-21 ##########################################################
> # requires objects: za.county (5-14 and 5-15)
> summary(lm(infections ~ pop + income + ipaddr + ufo2010,
+            data=za.county))

Call:
lm(formula = infections ~ pop + income + ipaddr + ufo2010, data = za.county)

Residuals:
     Min       1Q   Median       3Q      Max
-1077.10   -10.19    -3.60     2.89  1122.28

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.433e+01  4.855e+00   2.952  0.00318 **
pop          7.918e-04  7.945e-06  99.660  < 2e-16 ***
income      -3.283e-04  1.064e-04  -3.085  0.00205 **
ipaddr       2.070e-06  2.651e-07   7.807 7.99e-15 ***
ufo2010      4.375e-01  8.708e-02   5.024 5.34e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 65.6 on 3067 degrees of freedom
Multiple R-squared:  0.9438,    Adjusted R-squared:  0.9437
F-statistic: 1.287e+04 on 4 and 3067 DF,  p-value: < 2.2e-16
```

Listing 5-22: consist in calculating the variance inflation in the linear regression model. See figure 21 for output reference.

```
# Listing 5-22 ##########################################################
# requires objects: za.county (5-14 and 5-15)

model <- lm(infections ~ pop + income + ipaddr + ufo2010,
            data=za.county)
sqrt(vif(model))
```

Figure 21: Output for listing 5-22

```
> model <- lm(infections ~ pop + income + ipaddr + ufo2010,
+             data=za.county)
> sqrt(vif(model))
     pop   income   ipaddr  ufo2010
2.165458 1.038467 1.046051 2.115512
```

Listing 5-23: it normalizes the population to test the previous script. See figure 22 for output reference.

```
# Listing 5-23 ##########################################################
# requires objects: za.county (5-14 and 5-15)
za.county$za.by.pop <- za.county$infections/za.county$pop
za.county$ufo.by.pop <- za.county$ufo2010/za.county$pop
summary(lm(za.by.pop ~ ufo.by.pop, data=za.county))
```

Figure 22: Output for listing 5-23

```
> summary(lm(za.by.pop ~ ufo.by.pop, data=za.county))

Call:
lm(formula = za.by.pop ~ ufo.by.pop, data = za.county)

Residuals:
       Min         1Q     Median         3Q        Max
-0.0012548 -0.0003022 -0.0000563  0.0001912  0.0098066

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 7.050e-04  1.213e-05  58.108  < 2e-16 ***
ufo.by.pop  2.672e-01  6.956e-02   3.841 0.000125 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.0005793 on 3070 degrees of freedom
Multiple R-squared:  0.004784,  Adjusted R-squared:  0.004459
F-statistic: 14.76 on 1 and 3070 DF,  p-value: 0.0001248
```

Listing 5-24: consist in running the linear regression model using population as the main predictor for suspicious malware infections. See figure 23 for output reference

```
# Listing 5-24 ##########################################################
# requires objects: za.county (5-14 and 5-15)
summary(lm(infections ~ pop, data=za.county))
```

Figure 23: Output for listing 5-24

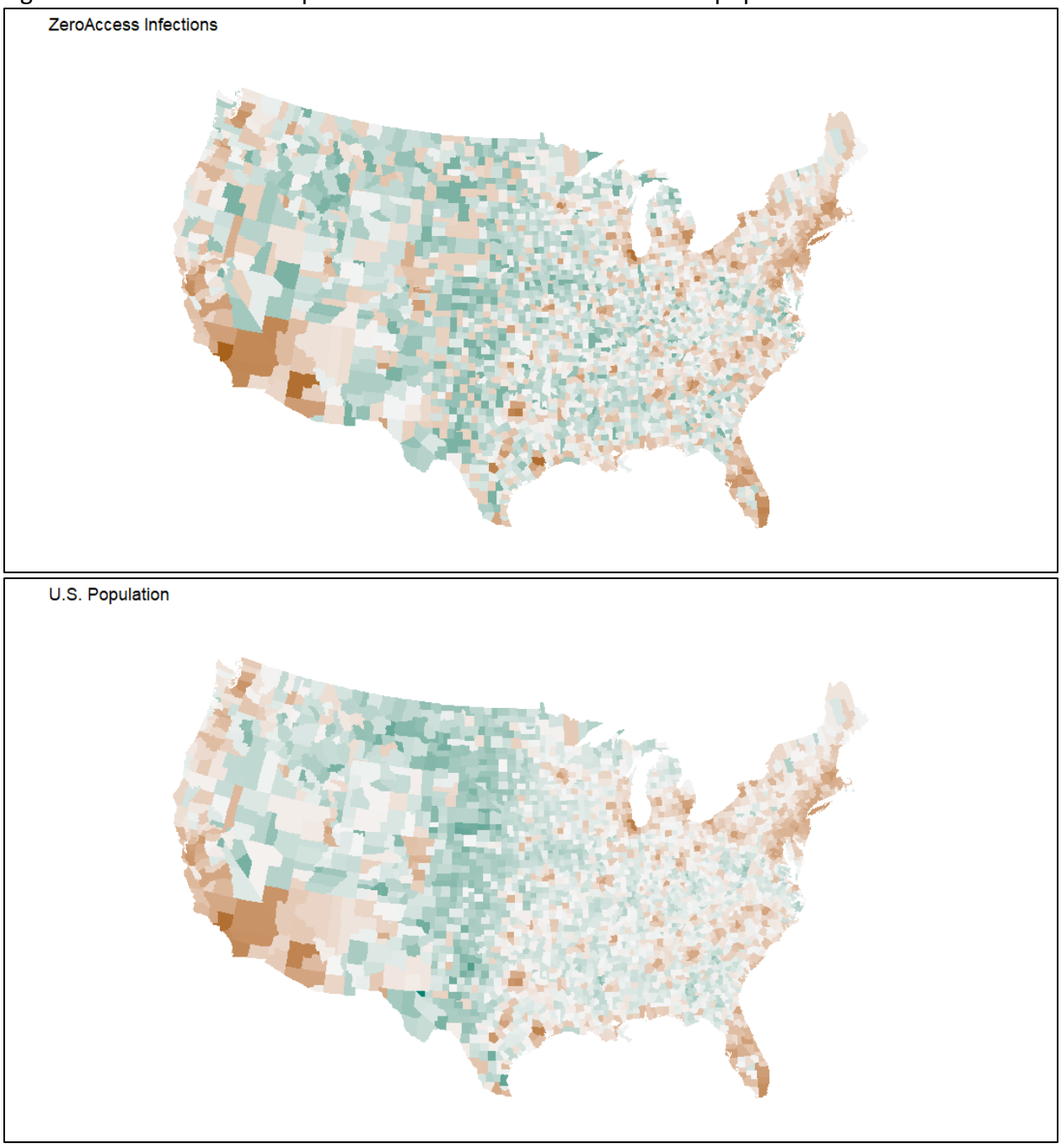
```
> # Listing 5-24 ##########################################################
> # requires objects: za.county (5-14 and 5-15)
> summary(lm(infections ~ pop, data=za.county))

Call:
lm(formula = infections ~ pop, data = za.county)

Residuals:
     Min       1Q   Median       3Q      Max
-1076.23    -9.17    -2.78     2.46  1106.30

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.173e-01  1.258e+00  -0.173    0.863
pop          8.317e-04  3.722e-06 223.478   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 66.54 on 3070 degrees of freedom
Multiple R-squared:  0.9421,    Adjusted R-squared:  0.9421
F-statistic: 4.994e+04 on 1 and 3070 DF,  p-value: < 2.2e-16
```

Listing for visual relationship between ZeroAccess infections and populations: This script is not in the book, but it is provided in the Wiley website. It generates a visual relationship between ZeroAccess infections and population. See figure 25 for output reference.

```
# Figure 5-10 in book ######################################################
# Code not in the book
# requires objects: za.county (5-14 and 5-15), colors (5-7)
base.county <- map_data("county")
map.county <- merge(base.county, za.county, all.x=T)
map.county <- map.county[with(map.county, order(group, order)), ]
# convert to log scale for the fill colors in the map
# wash over unknown counties by calling them average.
map.county$logpop <- ifelse(is.na(map.county$pop) | map.county$pop==0,
                            log10(mean(map.county$pop, na.rm=T)),
                            log10(map.county$pop))
map.county$logza <- ifelse(is.na(map.county$infections) |
map.county$infections==0,
                           log10(mean(map.county$infections, na.rm=T)),
                           log10(map.county$infections))

gpop <- ggplot(map.county, aes(x=long, y=lat, group=group, fill=logpop))
gpop <- gpop + geom_polygon(linetype=0)
gpop <- gpop + coord_map("polyconic")
gpop <- gpop + scale_fill_gradient2(low=colors[5], mid=colors[3],
                                    high=colors[1],
                                    midpoint=mean(map.county$logpop, na.rm=T),
                                    guide=F)
gpop <- gpop + ggtitle("U.S. Population")
gpop <- gpop + theme(axis.title=element_blank(),
                     axis.text=element_blank(),
                     axis.ticks=element_blank(),
                     panel.grid=element_blank(),
                     panel.background=element_blank())
print(gpop)

ginf <- ggplot(map.county, aes(x=long, y=lat, group=group, fill=logza))
ginf <- ginf + geom_polygon(linetype=0)
ginf <- ginf + coord_map("polyconic")
ginf <- ginf + scale_fill_gradient2(low=colors[5], mid=colors[3],
                                    high=colors[1],
                                    midpoint=mean(map.county$logza),
                                    guide=F)
ginf <- ginf + ggtitle("ZeroAccess Infections")
ginf <- ginf + theme(axis.title=element_blank(),
                     axis.text=element_blank(),
```

```
                        axis.ticks=element_blank(),
                    panel.grid=element_blank(),
                    panel.background=element_blank())
print(ginf)
```

Figure 24: visual relationship between ZeroAccess infections and populations



Appendix

Figure 25: Work signature in Rstudio.