```
######################################## Using R as a GIS ########################################

#### load the needed package
>library(GISTools)

#### load the data we need
>data(ewhaven)

#### what objects were loaded form the data
>ls()

#### what are the classes of the objects
>class(roads)
>class(blocks)
>class(breach)

#### let's look at more detailed information
>summary(breach)

#### or look at them in a longer version
>breach

#### let's look at each attribute of a layer solely
>breach$Long
>breach$Lat

#### lets try it with a line layer
>roads
>summary(roads)

#### lets see how the layers pop out
>plot(blocks)
>plot(roads, add=T, col="grey")
>plot(breach, col="red", add=T, pch=15)
```

let’s get rid of extra white spaces

>?par

>par(mar=c(0,0,1,0))

>plot(blocks)

>plot(roads, add=T, col="grey")

>plot(breach, col=”red”, add=T, pch=16)

add scale

>map.scale(534750, 152000, miles2ft(2), "Miles", 4, 0.5)

Adding the north arrow

>north.arrow(534750, 155000, miles2ft(0.25), col=”grey”)

adding title

>title("Breaches of the peace, New Haven, CT")

################################### Mapping Spatial Objects ###################################

clearing out the environment for next activity

>rm(list=ls())

if the package is still on you do not need to load it again. If not, please load it again

>library(GISTools)

load new data set

>data(“eorgia”)

let’s see what objectives of the data set

>ls()

let's figure out what variables georgia layer has

>names(Georgia)

let’s look at the countries of the state of Georgia

```
>plot(georgia)

#### let's make process of the mapping a little more fun
>plot(georgia, col="red")
>plot(eorgia, col=”red”, bg=”wheat”, lwd=3, border=”blue”)

### we can call one of the variables from the layer and just look at that
>georgia$PctRural

### another way to call a variable
data.frame(georgia)[,4]

######################################### More mapping tools ######################################

#### clearing out the environment for next activity
>rm(list=ls())

#### if the package is still on you do not need to load it again. If not, please load it again
>library(GISTools)

#### load a data
>data(ewhaven)

#### we can make a shade plot based on one of the information that is included in the layers. Here we are trying to make a color shaded map based on the proportion of vacant properties
>choropleth(blocks, blocks$P_VACANT)

######### creating a shading mP by defining categories
#### we are trying to define the shade color and number of categories
>shades <- auto.shading(blocks$P_VACANT, cols=brewer.pal(5, "Greens"))
#### drawing the shaded graph based on vacant properties but following the 5 categories
>choropleth(blocks, blocks$P_VACANT, shading=shades)
#### adding the legend for categories
>choro.legend(533000, 161000, shades, title="Prop. vacant")
#### adding a layer of points on the map
```

```
>plot(breach, col="red", pch=16, add=T, cex=0.6)
```

```
############################## Creating and importing GIS layers ###################################
```

```
#### importing our own data (x and Y) and turning it into spatial data file
#### we are generating some random points with minimum and maximum longitude and latitude of breach layer
>xmin <- min(breach$Long)
>xmax <- max(breach$Long)
>ymin <- min(breach$Lat)
>ymax <- max(breach$Lat)
>x <- runif(50, xmin, xmax)
>y <- runif(50, ymin, ymax)
# now plot the eandom points
>plot(x,y, add=T, col=”blue”)

#### combine the random points
>coords.tmp <- cbind(x,y)
#### give the points same projection as breach
>proj <- proj4string(breach)
#### turn them as a “SpatialPoints DataFrame
>points.spdf <- SpatialPointsDataFrame(coords.tmp, proj4string=CRS(proj), data=data.frame(cbind(x,y)))

#### clip the points and breach as the blocks polygon
# it is fine to get warnings
>points.clip <- gIntersection(points.spdf, blocks)
>breach.clip <- gIntersection(breach, blocks)

#### let’s plot what we have made
>plot(blocks)
>plot(breach.clip, col="red", add=T, cex=0.6, pch=16)
>plot(points.clip, col="blue", add=T, cex=0.6, pch=16)
```

```
########################## Mapping Raster data ##############################################

#### call the data. These are from the "sp" package
>data("meuse.grid")
>data("meuse")

#### let's investigate more about the layers
>class(meuse.grid)
## what are the variables in the data layer
>names(meuse.grid)

### "meuse.grid" is a data frame and we need to turn it to "SpatialPixelsDataFrame"
### the x and y are being used as latitude and longitude
>meuse.spatial <- SpatialPixelsDataFrame(points=meuse.grid[c("x", "y")],
                                data=meuse.grid)

#### let's plot them. But we are using spplot which is the plot from sp package
>spplot(meuse.spatial, "dist", col.regions=terrain.colors(20))
>spplot(meuse.spatial, "soil", col.regions=topo.colors(10))
>spplot(meuse.spatial, "ffreq", col.regions=heat.colors(20))
```