```
#################### igraph #############################

#### this package can be used for line and network analysis


## install and load the package 'igraph'
>install.packages("igraph")
>library(igraph)


#### for this example we will build up a simple graph ourselves
>g <- graph( c(1,2, 1,3, 2,3, 3,5), n=5)
### now let’s extract the nodes
>V(g)
### here we are going to extract the edges
>E(g)


### what are the differences between directed and undirected graphs?
>gDir <- graph(c(1,2, 1,3, 2,3, 3,5), n=5, dir=T)
>gUndir <- graph(c(1,2, 1,3, 2,3, 3,5), n=5, dir=F)
## let’s look at them simultaneously
>par(mfrow=c(1,2))
>plot(gDir)
>plot(gUndir)


#### let's have fun and play around with nodes and edges
### making a new undirected graph
>gUndir <- graph(c(1,2, 1,3, 2,3, 3,5), n=5, dir=F)
### here we color the vertices randomly
>V(gUndir)$color <- sample( c("red", "black"), vcount(gUndir), replace=T)
### now let's look at it
>plot(gUndir)


### you can, also, assign weight randomly to edges
>E(gUndir)$weight <- runif(ecount(gUndir))
```

```
>E(gUndir)$weight

### let's color the edges based on their weight
>E(gUndir)$color <- "grey"
>E(gUndir)[weight > 0.5]$color <- "red"
### let's see how it turns out
>plot(gUndir)

### here we want to show some other visualizations of a large networks
### different kinds of algorithms to make network graphs
>er_graph <- erdos.renyi.game(100, 2/100)
>par(mfrow=c(1,1))
>plot(er_graph, vertex.label=NA, vertex.size=3)

>ws_graph <- watts.strogatz.game(1, 100, 4, 0.05)
>plot(ws_graph, layout=layout.circle, vertex.label=NA, vertex.size=3)

>ba_graph <- barabasi.game(100)
>plot(ba_graph, vertex.label=NA, vertex.size=3)

#################### measuring network structure ###########################

### here we like to do some measurements, like average path length, network diameter, degree distribution, etc.
## so let's make a new graph
>roadnet <- graph(c(1,2, 1,3, 2,3, 3,5), n=5, dir=F)
>plot(roadnet)

### asking for shortest path from node 1 to 5
>shortest_paths(roadnet, from=1, to=5)

### let's find the most costly way to get from 1 to 3
### first we weight the nodes
>E(roadnet)$weight <- c(1,10,1,1)
```

```
## calculate the most expensive way
>shortest_paths(roadnet, from=1, to=5)


##### let's work with real data

#### install and load the packages
>install.packages(c("shp2graph","GISTools","raster"))
>library(shp2graph)
>library(GISTools)
>library(raster)

### the data we will work with is a railroad network from CA
## set the directory
>setwd("C:/Users/HP/OneDrive – University of Oklahoma/Oklahoma/outreach/shapefile")

##load the shapefile
>rails <- shapefile("California_Rail_Network.shp")
### let's see how it looks like
>plot(rails)

### the first check to see if the network has isolated subnetworks. We will use 'nt.connect' function.
### let's see what this function is
>?nt.connect
### apply it on our shapefile
>nt.connect(rails)
>plot(rt)
### here we have 68 self-connected parts

### let's just focus on the longest one from now on
>longest.rail <- nt.connect(rails)
>plot(longest.rail)
```

```
### we want to convert the line shapefile into nodelist and edgelist
## the function is 'readshpnw'. Let's explore this function
>?readshpnw
>rtNEL<-readshpnw(longest.rail, ELComputed=TRUE)

### let's look at the nodelist
>nodelist<-rtNEL[[2]]

### let's look at the edgelist
>edgelist<-rtNEL[[3]]

### now let's make a graph from these nodes and edges
# we are setting weight of each edge by adding the 4th element of the list
>railgraph <- nel2igraph(nodelist, edgelist, weight=rtNEL[[4]])
>plot(railgraph, vertex.size=0, vertex.label.cex=0.4)

### now let's find the shortest path from one node to another node (here as an example we are doing 139 to 401)
>shortpath <- shortest_paths(railgraph, from=139, to=401)
>shortpath
### let's see how long the shortest path between nodes 139 to 401 is
>shortpath.distances <- distances(railgraph, v=139, to=401)
>shortpath.distances

### let's see how this short path looks
>E(railgraph)$color <- "blue"
>E(railgraph, path=shortpath$vpath[[1]])$color <- "red"
>plot.igraph(railgraph, vertex.label=NA, vertex.size=0)
```