Social Network Analysis with R *

Yanchang Zhao

http://www.RDataMining.com

Tutorial on Machine Learning with R

The Melbourne Data Science Week 2017

1 June 2017

*Chapter 11: Social Network Analysis, in *R and Data Mining: Examples and Case Studies*. http://www.rdatamining.com/docs/RDataMining-book.pdf

Contents

Network and Graph

- Nodes, vertices or entities
- Edges, links or relationships
- Network analysis, graph mining
- Link prediction, community/group detection, entity resolution, recommender system, information propogation modeling

Graph Databases

- Neo4j: `https://neo4j.com`
- Giraph on Hadoop: `http://giraph.apache.org`
- GraphX on Spark: `http://spark.apache.org/graphx/`

Social Network Analysis

- Graph construction
- Graph query
- Centrality measures
- Graph visualization
- Clustering and community detection

Contents

Graph Construction

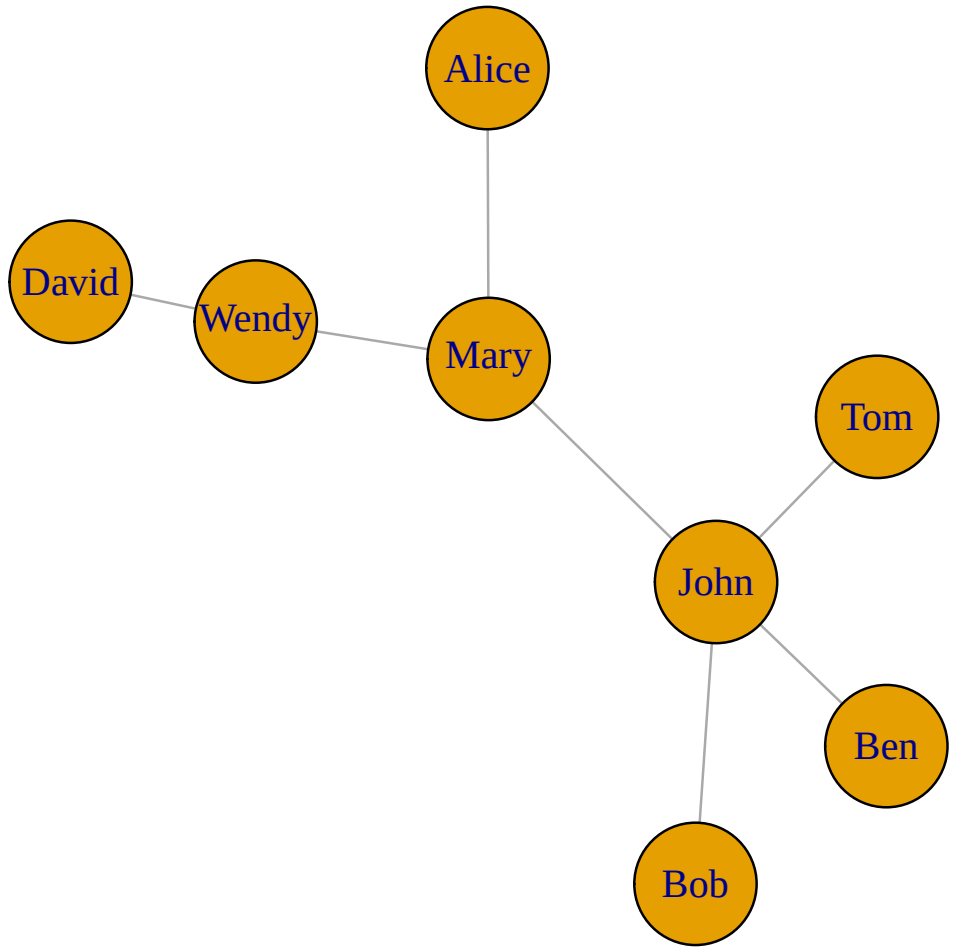
- Tom, Ben, Bob and Mary are friends of John.
- Alice and Wendy are friends of Mary.
- Wendy is a friend of David.

```
library(igraph)
# nodes
nodes <- data.frame(
  name = c("Tom","Ben","Bob","John","Mary","Alice","Wendy","David"),
  gender = c("M", "M", "M", "M", "F", "F", "F", "M"),
  age =    c( 16,  30,  42,  29,  26,  32,  18,  22)
)
# relations
edges <- data.frame(
  from = c("Tom", "Ben",  "Bob", "Mary", "Alice", "Wendy", "Wendy"),
  to   =  c("John", "John", "John", "John","Mary", "Mary", "David")
)
# build a graph object
g <- graph.data.frame(edges, directed=F, vertices=nodes)
```

Contents

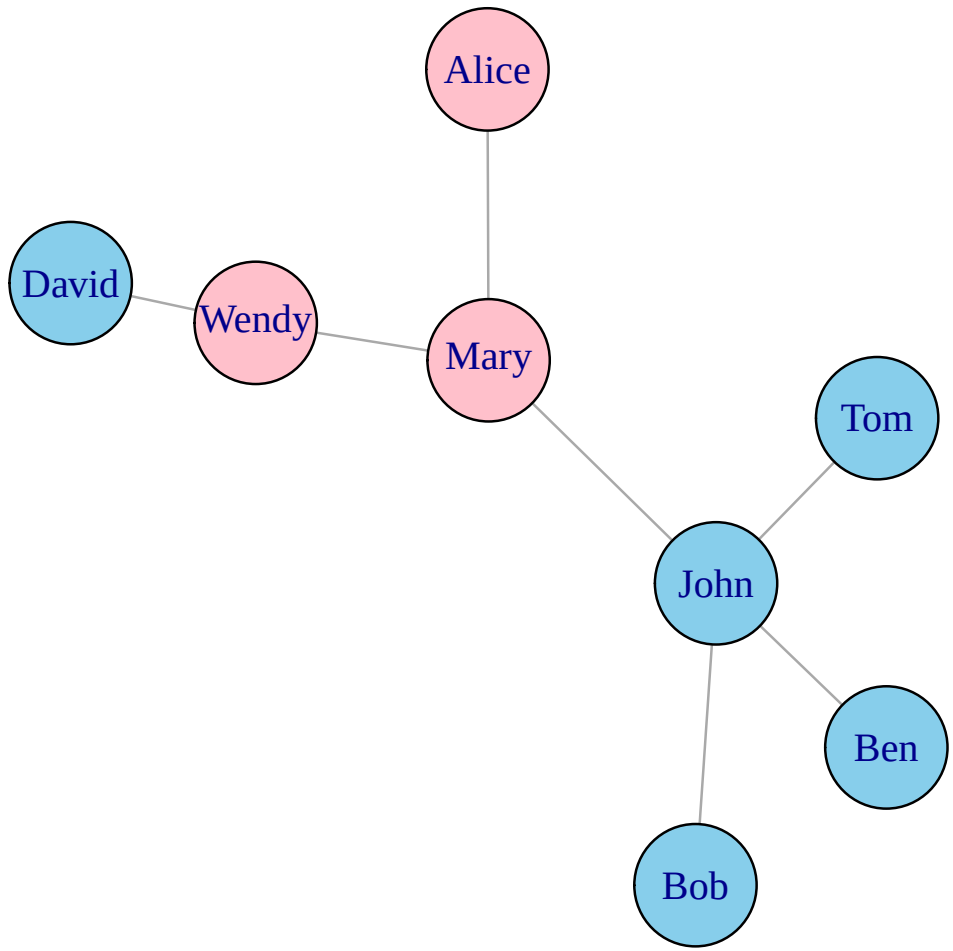
Graph Visualization

```
layout1 <- g %>% layout_nicely() ## save layout for reuse
g %>% plot(vertex.size=30, layout=layout1)
```



Graph Visualization (cont.)

```
## use blue for male and pink for female
colors <-  ifelse(V(g)$gender=="M", "skyblue", "pink")
g %>% plot(vertex.size=30, vertex.color=colors, layout=layout1)
```



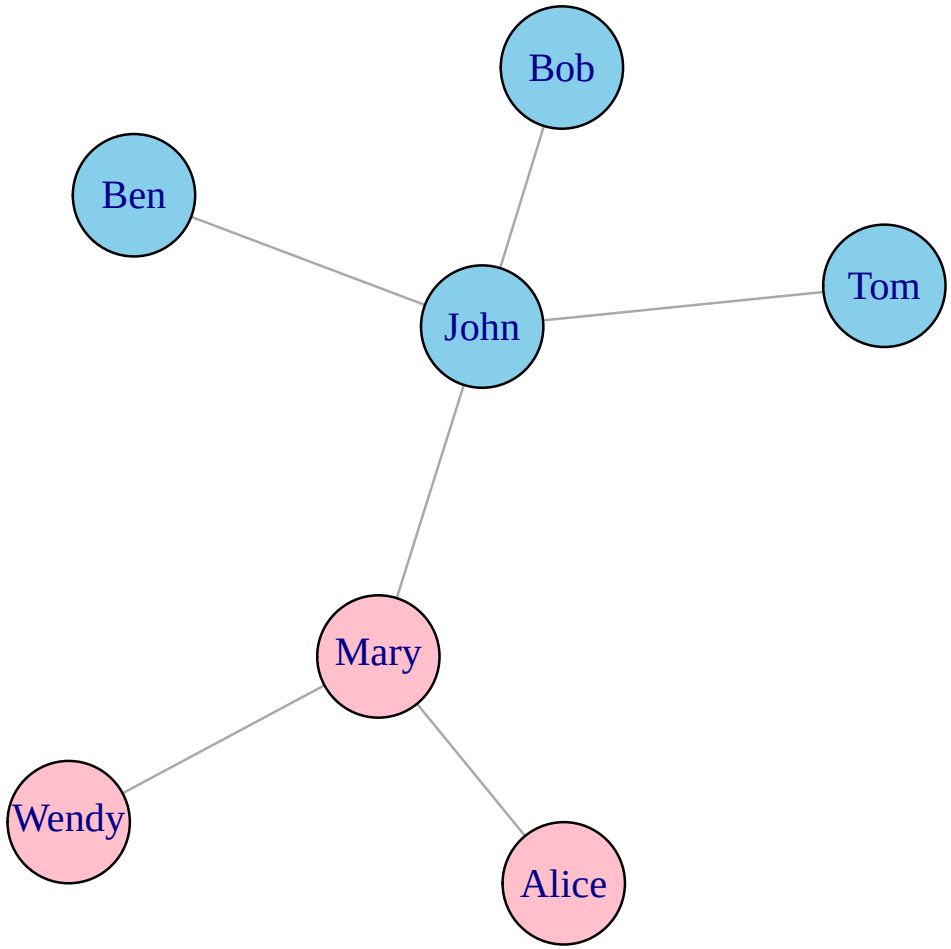
Contents

Graph Query

```
## nodes
V(g)
## + 8/8 vertices, named:
## [1] Tom   Ben   Bob   John  Mary  Alice Wendy David

## edges
E(g)
## + 7/7 edges (vertex names):
## [1] Tom  --John  Ben  --John  Bob  --John  John --Mary
## [5] Mary --Alice Mary --Wendy Wendy--David

## immediate neighbors (friends) of John
friends <- ego(g,order=1,nodes="John",mindist=1)[[1]] %>% print()
## + 4/8 vertices, named:
## [1] Tom  Ben  Bob  Mary

## female friends of John
friends[friends$gender == "F"]
## + 1/8 vertex, named:
## [1] Mary
```

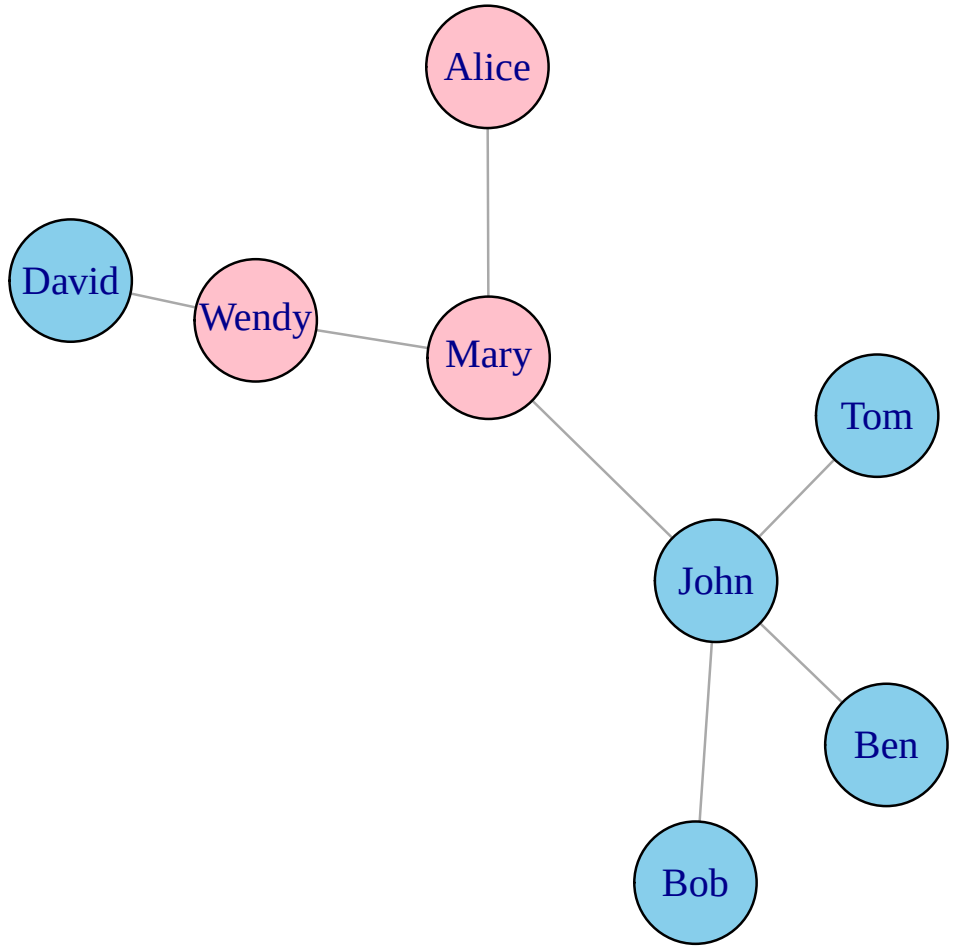
Graph Query (cont.)

```
## 1- and 2-order neighbors (friends) of John
g2 <- make_ego_graph(g, order=2, nodes="John")[[1]]
g2 %>% plot(vertex.size=30, vertex.color=colors)
```



Contents

Friendship Graph



Centrality Measures

- Degree: the number of adjacent edges; indegree and outdegree for directed graphs
- Closeness: the inverse of the average length of the shortest paths to/from all other nodes
- Betweenness: the number of shortest paths going through a node

```
degree <- g %>% degree() %>% print()
##   Tom   Ben   Bob  John  Mary Alice Wendy David
##     1     1     1     4     3     1     2     1

closeness <- g %>% closeness() %>% round(2) %>% print()
##   Tom   Ben   Bob  John  Mary Alice Wendy David
##  0.06  0.06  0.06  0.09  0.09  0.06  0.07  0.05

betweenness <- g %>% betweenness() %>% print()
##   Tom   Ben   Bob  John  Mary Alice Wendy David
##     0     0     0    15    14     0     6     0
```

Centrality Measures (cont.)

- Eigenvector centrality: the values of the first eigenvector of the graph adjacency matrix
- Transivity, a.k.a clustering coefficient, measures the probability that the adjacent nodes of a node are connected.

```
eigenvector <- evcent(g)$vector %>% round(2) %>% print()
##   Tom   Ben   Bob  John  Mary Alice Wendy David
##  0.45  0.45  0.45  1.00  0.85  0.38  0.48  0.22

transitivity <- g %>% transitivity(type="local") %>% print()
## [1] NaN NaN NaN   0   0 NaN   0 NaN
```

Contents

Static Network Visualization

- Static network visualization
- Fast in rendering big graphs
- For very big graphs, the most efficient way is to save visualization result into a file, instead of directly to screen.
- Save network diagram into files: `pdf()`, `bmp()`, `jpeg()`, `png()`, `tiff()`

```
library(igraph)

## plot directly to screen when graph is small
plot(g)

## for big graphs, save visualization to a PDF file
pdf("mygraph.pdf")
plot(g)
graphics.off()  ## or dev.off()
```

Interactive Network Visualization

- Coordinates of other nodes are not adjusted when moving a node.
- Can be slow when rendering big graphs
- Save network diagram into files: `visSave()`, `visExport()`

```
visIgraph(g, idToLabel=T) %>%
  ## highlight nodes connected to a selected node
  visOptions(highlightNearest=T) %>%
  ## use different icons for different types (groups) of nodes
  visGroups(groupname="person", shape="icon",
            icon=list(code="f007")) %>%
  ... %>%
  ## use FontAwesome icons
  addFontAwesome() %>%
  ## add legend of nodes
  visLegend() %>%
  ## to save to file
  visSave(file = "network.html")
```

Interactive Network Visualization (cont.)

- Dynamically adjusting coordinates for better visualization
- Very slow when rendering big graphs

```
x <- toVisNetworkData(g)
visNetwork(nodes=x$nodes, edges=x$edges)%>%
  ## use different icons for different types (groups) of nodes
  visGroups(groupname="person", shape="icon",
            icon=list(code="f007")) %>%
  ... %>%
  ## use FontAwesome icons
  addFontAwesome() %>%
  ## add legend of nodes
  visLegend()
```

Load Graph Data

```
## download graph data
url <- "http://www.rdatamining.com/data/graph.rdata"
download.file(url, destfile="./data/graph.rdata")
```

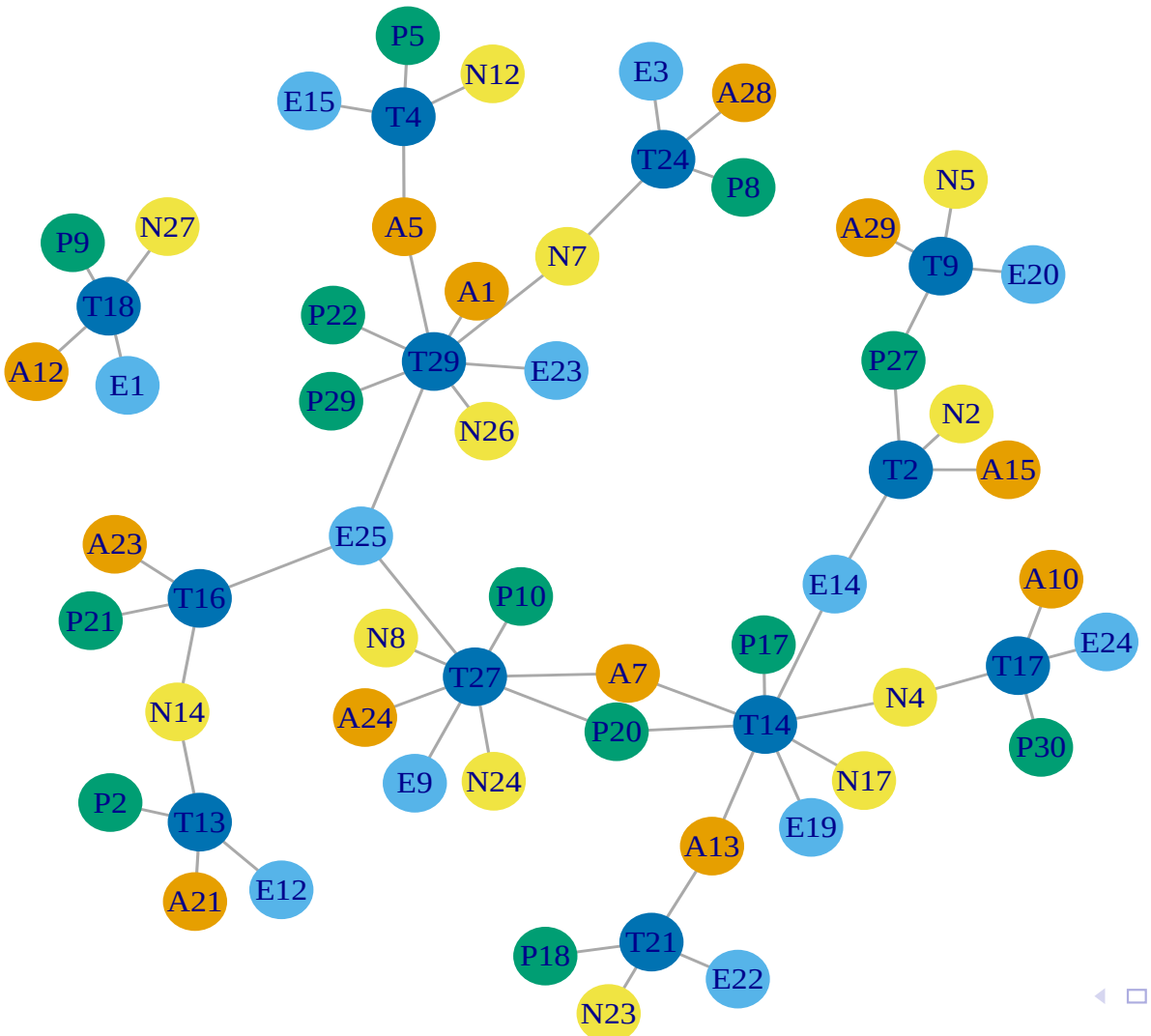
```
library(igraph)
# load graph data into R
# what will be loaded: g, nodes, edges
load("./data/graph.rdata")
```

Build a Graph

```
head(nodes, 3)
##   name type
## 1   T9  tid
## 2  T24  tid
## 3  T13  tid

head(edges, 3)
##   from  to
## 1   T9 P27
## 2  T24  P8
## 3  T13  P2

## buid a graph object
g <- graph.data.frame(edges, directed=F, vertices=nodes)
g
## IGRAPH UN-B 61 60 --
## + attr: name (v/c), type (v/c)
## + edges (vertex names):
##  [1] T9 --P27 T24--P8  T13--P2  T27--P10 T29--P29 T2 --P27
##  [7] T16--P21 T27--P20 T17--P30 T14--P20 T29--P22 T14--P17
## [13] T21--P18 T18--P9  T4 --P5  T9 --A29 T24--A28 T13--A21
```

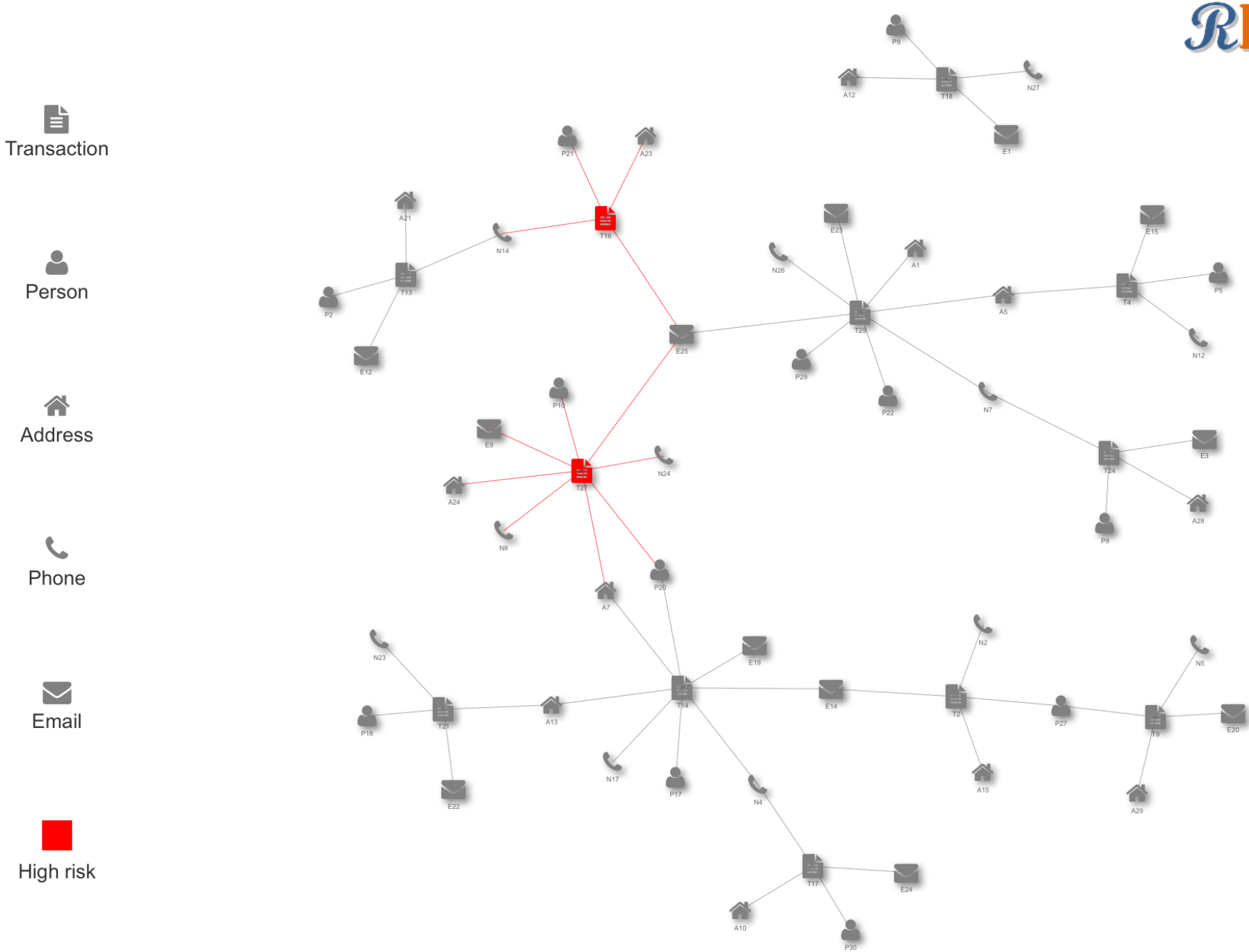
Example of Static Network Visualization

```
library(igraph)
plot(g, vertex.size=12, vertex.label.cex=0.7,
     vertex.color=as.factor(V(g)$type), vertex.frame.color=NA)
```



Example of Interactive Network Visualization

```
library(visNetwork)
V(g)$group <- V(g)$type
## visualization
data <- toVisNetworkData(g)
visNetwork(nodes=data$nodes, edges=data$edges) %>%
visGroups(groupname="tid",shape="icon",icon=list(code="f15c")) %>%
visGroups(groupname="person",shape="icon",icon=list(code="f007")) %>%
visGroups(groupname="addr",shape="icon",icon=list(code="f015")) %>%
visGroups(groupname="phone",shape="icon",icon=list(code="f095")) %>%
visGroups(groupname="email",shape="icon",icon=list(code="f0e0")) %>%
addFontAwesome() %>%
visLegend()
```


Transaction
Person
Address
Phone
Email
High risk
P9
A12
T18
N27
E1
P21
A23
A21
T16
N14
T13
P2
E12
E23
A1
N26
E15
P5
A5
T4
N12
T23
E25
P29
P22
N7
P10
E9
N24
T24
E3
A24
T27
A28
N8
P8
P20
A7
N23
E19
N2
N5
P18
T21
A13
T14
E14
T2
P27
T9
E20
N17
P17
N4
A15
A29
E22
T17
E24
A10
P30

Contents

R Packages

- Network analysis: *igraph*, *sna*, *statnet*
- Network visualization: *visNetwork*
- Interface with graph databases: *RNeo4j*

Package *igraph* †

- `V(g)`, `E(g)`: nodes and edges of graph `g`
- `degree`, `betweenness`, `closeness`, `transitivity`: various centrality scores
- `neighborhood`: neighborhood of graph vertices
- `cliques`, `largest.cliques`, `maximal.cliques`, `clique.number`: find cliques, ie. complete subgraphs
- `clusters`, `no.clusters`: maximal connected components of a graph and the number of them
- `fastgreedy.community`, `spinglass.community`: community detection
- `cohesive.blocks`: calculate cohesive blocks
- `induced.subgraph`: create a subgraph of a graph (*igraph*)
- `read.graph`, `write.graph`: read and writ graphs from and to files of various formats

† https://cran.r-project.org/package=igraph

Contents

Wrap Up

- Package *igraph* and *sna*
  - Static visualization
  - Can visualize nodes with shapes, images and icons
  - Visualise very large graph
  - Support network analysis and graph mining
- Package *visNetwork*
  - Interactive visualization
  - Can visualize nodes with shapes, images and icons
  - Image rendering can be very slow for large graphs
  - Designed for visualization only, and does not support network analysis and graph mining

Contents

Further Readings

- Social network analysis (SNA)

  `https://en.wikipedia.org/wiki/Social_network_analysis`
- igraph – a network analysis package, supporting R, Python and C/C++

  `http://igraph.org`
- sna – an R package for social network analysis

  `https://cran.r-project.org/web/packages/sna/index.html`
- statnet – software tools for the analysis, simulation and visualization of network data; also available as an R package

  `http://www.statnet.org`
- visNetwork – an R package for network visualization

  `http://datastorm-open.github.io/visNetwork/`

Online Resources

- Chapter 11 – Social Network Analysis, in book *R and Data Mining: Examples and Case Studies*

  `http://www.rdatamining.com/docs/RDataMining-book.pdf`
- RDataMining Reference Card

  `http://www.rdatamining.com/docs/RDataMining-reference-card.pdf`
- Online documents, books and tutorials

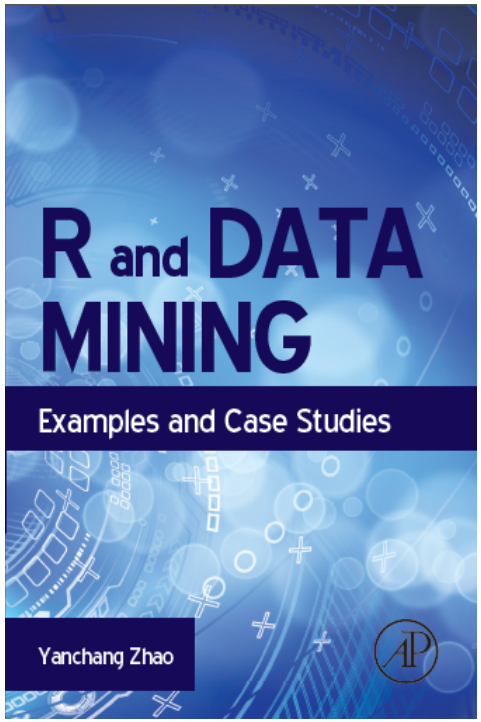
  `http://www.rdatamining.com/resources/onlinedocs`
- Free online courses

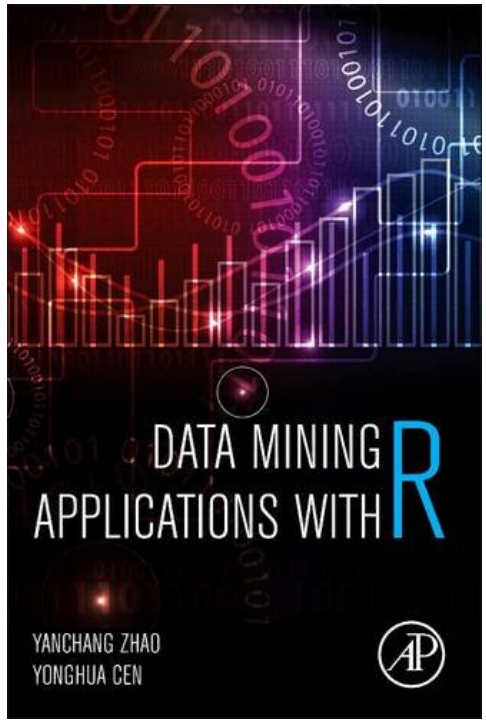
  `http://www.rdatamining.com/resources/courses`
- RDataMining Group on LinkedIn (24,000+ members)

  `http://group.rdatamining.com`
- Twitter (3,000+ followers)

  @RDataMining

The End





Thanks!

Email: yanchang(at)RDataMining.com
Twitter: @RDataMining