THE OHIO STATE UNIVERSITY
FISHER COLLEGE OF BUSINESS

BUSMGT 7331: Descriptive Analytics and Visualization
Week 5

Network Visualization

Hyunwoo Park
Fisher College of Business
The Ohio State University

Textbooks for this week

- The following two booklets were created by Professor Katherine Ognyanova at Rutgers University.
- [Og2016] Network analysis with R and igraph
  - http://kateto.net/networks-r-igraph
  - PDF: http://www.kateto.net/wp-content/uploads/2016/01/NetSciX_2016_Workshop.pdf
- [Og2018] Static and dynamic visualization with R
  - http://kateto.net/network-visualization
  - PDF: http://kateto.net/wp-content/uploads/2018/06/Polnet%202018%20R%20Network%20Visualization%20Workshop.pdf
- Citations
  - Ognyanova, K. (2016) Network analysis with R and igraph: NetSci X Tutorial. Retrieved from www.kateto.net/networks-r-igraph.
  - Ognyanova, K. (2018) *Network visualization with R.* Retrieved from www.kateto.net/network-visualization.

Technical readings for this week

- Introduction to Network Analysis in R
  https://www.jessesadler.com/post/network-analysis-with-r/
- Introduction of tidygraph package
  https://www.data-imaginist.com/2017/introducing-tidygraph/
- Introduction of ggraph package
  https://www.data-imaginist.com/2017/announcing-ggraph/
- Vignettes of ggraph package
  - https://cran.r-project.org/web/packages/ggraph/vignettes/Nodes.html
  - https://cran.r-project.org/web/packages/ggraph/vignettes/Edges.html
  - https://cran.r-project.org/web/packages/ggraph/vignettes/Layouts.html

Packages to install

```
1  install.packages("tidygraph")
2  install.packages("ggraph")
```

- If you install these two packages, igraph package will be automatically installed as a dependency to ggraph. If igraph is not installed for any reason, you should install it separately.

Anatomy of Networks

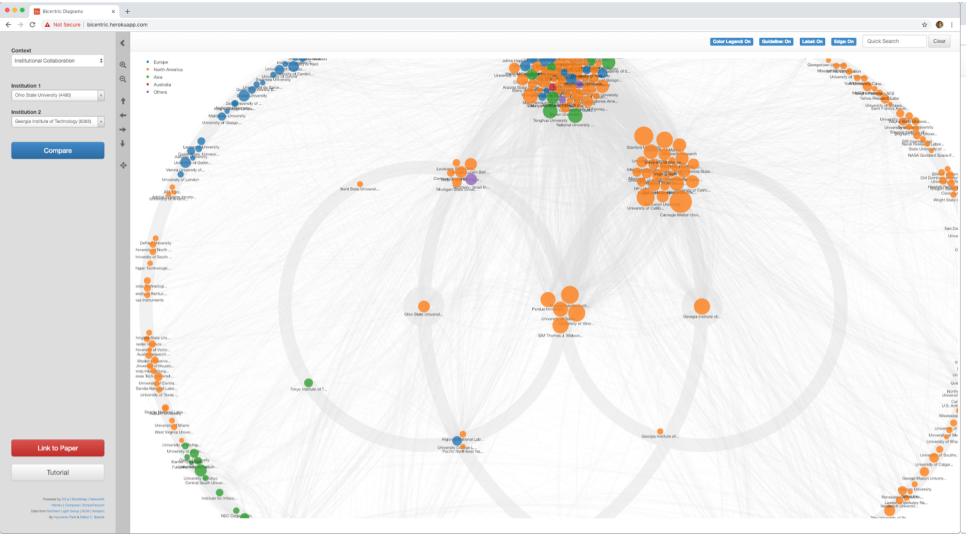
Reading

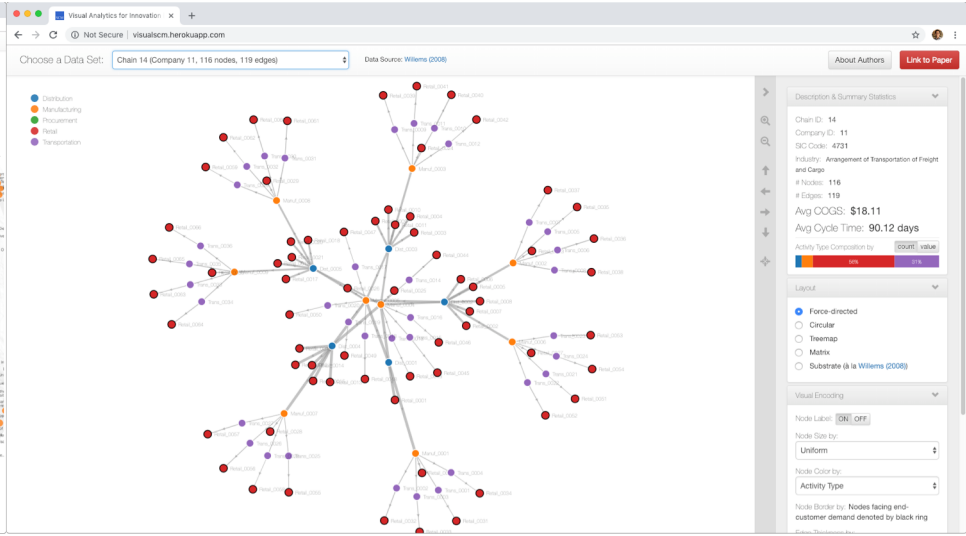
- Og2016 Section 2: Networks in igraph
- Og2016 Section 3: Reading network data from files
- Og2016 Section 4: Turning networks into igraph objects

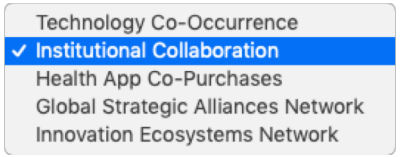
Network theory

- Network is a set of entities connected to each other via relationships.
- Many natural and social phenomena can be thought of as networks.
- In mathematics, graph theory established a long time ago the basic framework to conceptualize and analyze a network. I will use network and graph interchangeably.
- Around 1970s, sociologists applied the network lens to analyze relationships in society and popularized the use of social network analysis (SNA) in their studies.
- A few starting points for getting the overview of network analysis.
  - https://en.wikipedia.org/wiki/Network_theory
  - https://en.wikipedia.org/wiki/Social_network_analysis

My own examples of network visualization







Figures from Bicentric Diagrams and Visual SCM

Key concepts in graph theory

- A network is a set of entities and a set of relationships.
- The entities are called “vertices”, “nodes”, “points”.
  The relationships are called “edges”, “links”, “arcs”, “lines”.
- In graph theory, vertex and edge are most commonly used.
  Thus, the mathematical notation for a graph is: $G = \{V, E\}$

Understanding data structure for network

- One key difference between a network data and a usual tabular data is that a network data (typically) consists of two tables as a set.
- One table contains information about nodes and the other about edges.
- The core data necessary to create a network is the edges table because it describes the relationship between entities and the nodes table provides additional information about entities in the network.
- Because of this dual-table structure of network data, software that analyzes networks has had its own niche. The igraph package has been one of the most widely used network analysis tool in both R and Python.
- Recently, thanks to tidygraph and ggraph packages developed by Thomas Lin Pederson, we can explore networks in a tidy way and visualize in a ggplot way.

Creating igraph object

- There is a sample data set that comes with ggraph package called highschool and flare. To learn more about those datasets, type ?highschool and ?flare.

```
grr <- make_ring(10)
grt <- graph_from_literal(A-B-C,A-D-E,B-D)
grh <- graph_from_data_frame(highschool)
grf <- graph_from_data_frame(flare$edges, vertices=flare$vertices)
```

```
> grr
IGRAPH bd61722 U--- 10 10 -- Ring graph
+ attr: name (g/c), mutual (g/l), circular (g/l)
+ edges from bd61722:
 [1] 1-- 2 2-- 3 3-- 4 4-- 5 5-- 6 6-- 7 7-- 8 8-- 9 9--10 1--10
```

```
> grt
IGRAPH 618a235 UN-- 5 5 --
+ attr: name (v/c)
+ edges from 618a235 (vertex names):
[1] A--B A--D B--C B--D D--E
```

```
> grf
IGRAPH e543cc0 DN-- 252 251 --
+ attr: name (v/c), size (v/n), shortName (v/c)
+ edges from e543cc0 (vertex names):
[1] flare.analytics.cluster->flare.analytics.cluster.AgglomerativeCluster
[2] flare.analytics.cluster->flare.analytics.cluster.CommunityStructure
[3] flare.analytics.cluster->flare.analytics.cluster.HierarchicalCluster
[4] flare.analytics.cluster->flare.analytics.cluster.MergeEdge
[5] flare.analytics.graph  ->flare.analytics.graph.BetweennessCentrality
[6] flare.analytics.graph  ->flare.analytics.graph.LinkDistance
[7] flare.analytics.graph  ->flare.analytics.graph.MaxFlowMinCut
[8] flare.analytics.graph  ->flare.analytics.graph.ShortestPaths
+ ... omitted several edges
```

```
> grh
IGRAPH f712c0f DN-- 70 506 --
+ attr: name (v/c), year (e/n)
+ edges from f712c0f (vertex names):
  [1] 1 ->14 1 ->15 1 ->21 1 ->54 1 ->55 2 ->21 2 ->22 3 ->9  3 ->15 4 ->5  4 ->18 4 ->19 4 ->43
 [14] 5 ->19 5 ->43 6 ->13 6 ->20 6 ->22 7 ->17 8 ->14 8 ->17 9 ->12 9 ->20 9 ->21 9 ->22 9 ->51
 [27] 11->19 11->50 11->52 11->53 12->20 12->21 12->22 13->17 13->20 13->21 13->22 14->21 14->22
 [40] 15->20 16->18 16->41 16->43 17->7  17->8  18->11 18->16 18->19 19->4  19->11 19->16 19->18
 [53] 19->27 20->6  20->12 20->21 20->22 20->38 21->22 21->51 21->54 21->55 22->20 22->21 22->38
 [66] 22->51 23->40 23->43 23->50 23->52 23->53 23->60 23->62 23->65 23->68 24->51 26->32 26->35
 [79] 26->36 26->40 26->41 26->42 27->18 27->37 27->40 28->38 28->39 29->13 29->38 30->35 30->48
 [92] 31->10 31->37 31->40 32->26 32->33 32->43 32->62 33->36 33->41 33->42 33->43 34->39 34->46
+ ... omitted several edges
```

Understanding igraph object

- Graph-level attributes
  - U/D: Undirected vs directed graph
  - N/-: Named or not (i.e., nodes table has a column called “name”)
  - W/-: Weighted or not (i.e., edges table has a column called “weight”)
  - B/-: Bipartite or not (i.e., nodes table has a column called “type”)
  - Number of nodes / number of edges
- Other attributes
  - (g/c): graph-level character attribute
  - (v/c): node-level (vertex-level) character attribute
  - (e/c): edge-level character attribute
  - (g/n): graph-level numeric attribute
  - (v/n): node-level (vertex-level) numeric attribute
  - (e/n): edge-level numeric attribute

```
> grt
IGRAPH 618a235 UN-- 5 5 --
+ attr: name (v/c)
+ edges from 618a235 (vertex names):
[1] A--B A--D B--C B--D D--E
```

Using tidygraph to manipulate a graph in a tidy way

- Reading: https://www.data-imaginist.com/2017/introducing-tidygraph/

```
1  tgrf <- as_tbl_graph(grf)
2  tgrf
3  tgrf %>% arrange(-size)
4  as_tibble(tgrf %>% activate(edges))
```

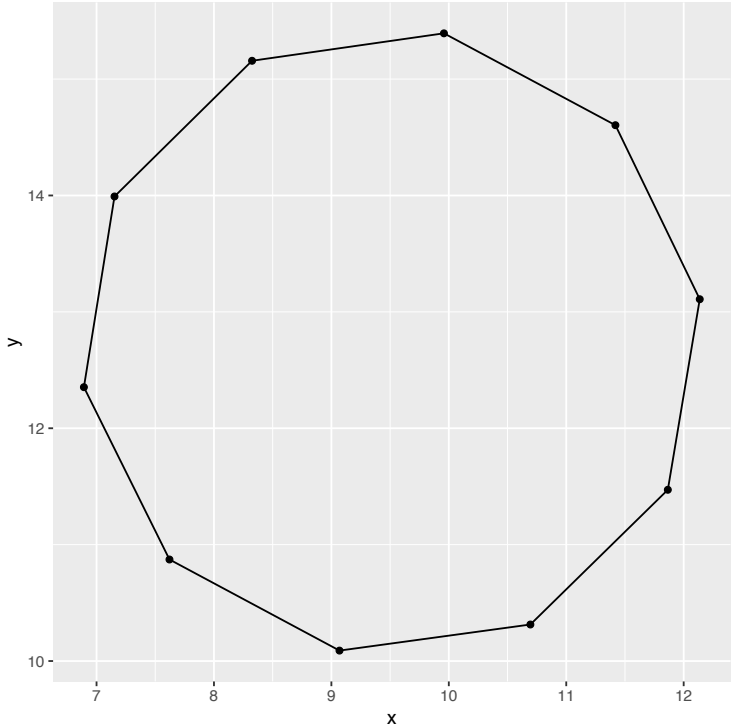
```
> tgrf
# A tbl_graph: 252 nodes and 251 edges
#
# A rooted tree
#
# Node Data: 252 x 3 (active)
  name                                          size shortName
  <chr>                                        <dbl> <chr>
1 flare.analytics.cluster.AgglomerativeCluster  3938 AgglomerativeCluster
2 flare.analytics.cluster.CommunityStructure    3812 CommunityStructure
3 flare.analytics.cluster.HierarchicalCluster   6714 HierarchicalCluster
4 flare.analytics.cluster.MergeEdge              743 MergeEdge
5 flare.analytics.graph.BetweennessCentrality   3534 BetweennessCentrality
6 flare.analytics.graph.LinkDistance            5731 LinkDistance
# ... with 246 more rows
#
# Edge Data: 251 x 2
   from    to
  <int> <int>
1   221     1
2   221     2
3   221     3
# ... with 248 more rows
```

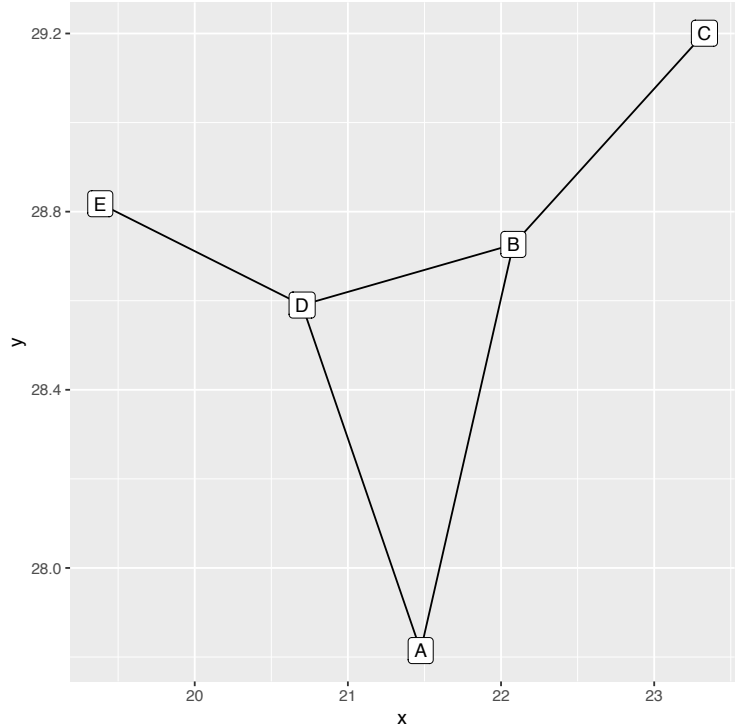
```
> tgrf %>% arrange(-size)
# A tbl_graph: 252 nodes and 251 edges
#
# A rooted tree
#
# Node Data: 252 x 3 (active)
  name                          size shortName
  <chr>                        <dbl> <chr>
1 flare.vis.axis.Axis          24593 Axis
2 flare.util.Strings           22026 Strings
3 flare.vis.legend.Legend      20859 Legend
4 flare.vis.data.Data          20544 Data
5 flare.animate.Transitioner   19975 Transitioner
6 flare.vis.data.DataList      19788 DataList
# ... with 246 more rows
#
# Edge Data: 251 x 2
   from    to
  <int> <int>
1   221    81
2   221    85
3   221    46
# ... with 248 more rows
```

```
> as_tibble(tgrf %>% activate(edges))
# A tibble: 251 x 2
    from    to
   <int> <int>
 1   221     1
 2   221     2
 3   221     3
 4   221     4
 5   222     5
 6   222     6
 7   222     7
 8   222     8
 9   222     9
10   223    10
# ... with 241 more rows
```

Simple network visualization

```
1  ggraph(grr) + geom_node_point() + geom_edge_link()
2  ggraph(grt) + geom_edge_link() + geom_node_label(aes(label=name))
```





Graph represented in adjacency matrix

```
1  as_adj(grr)
2  as_adj(grt)
```

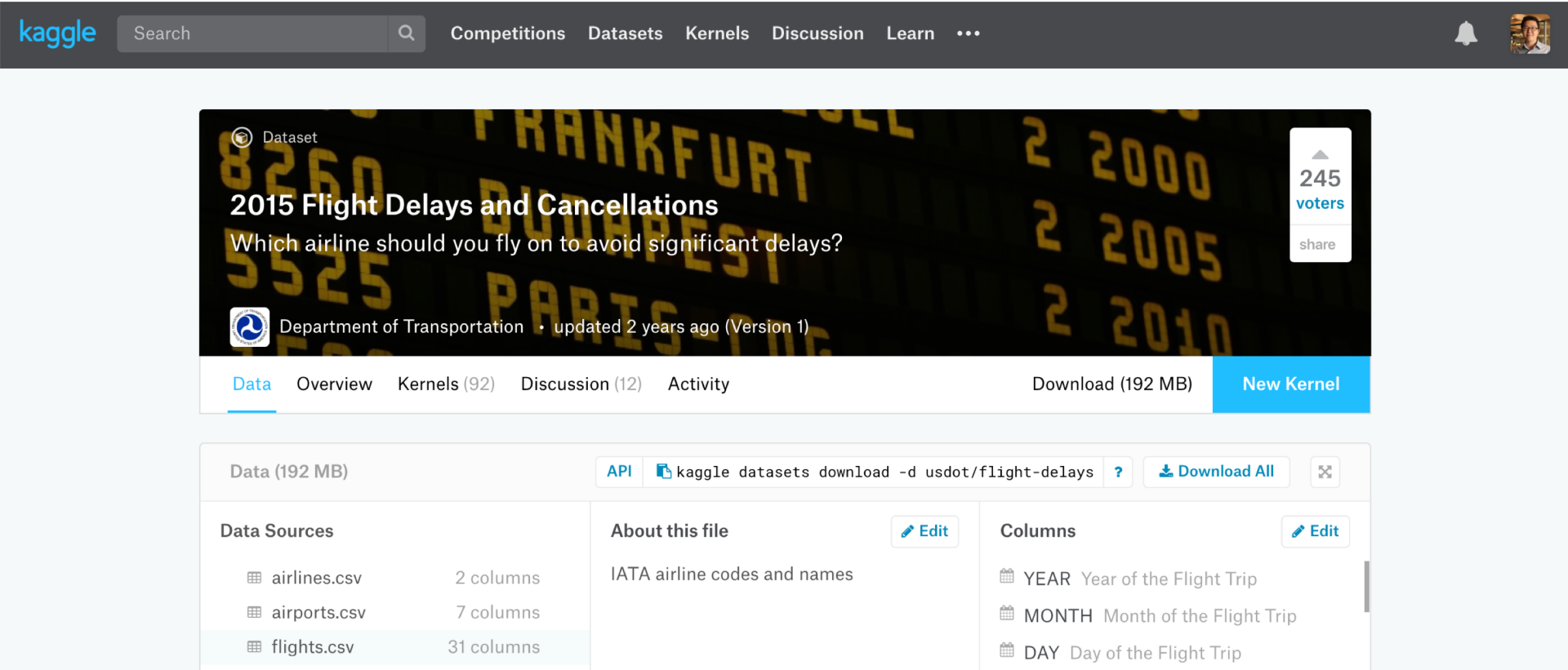
```
> as_adj(grr)
10 x 10 sparse Matrix of class "dgCMatrix"

 [1,] . 1 . . . . . . . 1
 [2,] 1 . 1 . . . . . . .
 [3,] . 1 . 1 . . . . . .
 [4,] . . 1 . 1 . . . . .
 [5,] . . . 1 . 1 . . . .
 [6,] . . . . 1 . 1 . . .
 [7,] . . . . . 1 . 1 . .
 [8,] . . . . . . 1 . 1 .
 [9,] . . . . . . . 1 . 1
[10,] 1 . . . . . . . 1 .
```

```
> as_adj(grt)
5 x 5 sparse Matrix of class "dgCMatrix"
  A B C D E
A . 1 . 1 .
B 1 . 1 1 .
C . 1 . . .
D 1 1 . . 1
E . . . 1 .
```

Dataset for this week

- https://www.kaggle.com/usdot/flight-delays



Wrangling for creating a network data

```
1   airports <- read_csv("data/airports.csv")
2   flights <- read_csv("data/flights.csv.zip")
3
4   nodes <- airports %>% rename(name=IATA_CODE, airport=AIRPORT, city=CITY,
    state=STATE, country=COUNTRY, lat=LATITUDE, lon=LONGITUDE)
5   edges <- flights %>% group_by(ORIGIN_AIRPORT, DESTINATION_AIRPORT) %>%
6     count() %>% ungroup() %>%
7     semi_join(airports, by=c("ORIGIN_AIRPORT"="IATA_CODE")) %>%
8     semi_join(airports, by=c("DESTINATION_AIRPORT"="IATA_CODE")) %>%
9     mutate(from=ORIGIN_AIRPORT, to=DESTINATION_AIRPORT, weight=n) %>%
10    select(from, to, weight)
11
12  write_csv(nodes, "data/nodes.csv")
13  write_csv(edges, "data/edges.csv")
```

Importing network data

```
1  nodes <- read_csv("data/nodes.csv")
2  edges <- read_csv("data/edges.csv")
3  tgra <- tbl_graph(nodes=nodes, edges=edges)
4  as.igraph(tgra)
5  as_adj(as.igraph(tgra))
```

```
> tgra
# A tbl_graph: 322 nodes and 4693 edges
#
# A directed simple graph with 1 component
#
# Node Data: 322 x 7 (active)
  name  airport                          city        state country   lat    lon
  <chr> <chr>                            <chr>       <chr> <chr>   <dbl>  <dbl>
1 ABE   Lehigh Valley International Airport Allentown   PA    USA      40.7  -75.4
2 ABI   Abilene Regional Airport           Abilene     TX    USA      32.4  -99.7
3 ABQ   Albuquerque International Sunport  Albuquerque NM    USA      35.0 -107.
4 ABR   Aberdeen Regional Airport          Aberdeen    SD    USA      45.4  -98.4
5 ABY   Southwest Georgia Regional Airport Albany      GA    USA      31.5  -84.2
6 ACK   Nantucket Memorial Airport         Nantucket   MA    USA      41.3  -70.1
# ... with 316 more rows
#
# Edge Data: 4,693 x 3
   from    to weight
  <int> <int>  <dbl>
1     1    21    898
2     1    94    711
3     1   229    665
# ... with 4,690 more rows
```

```
> as.igraph(tgra)
IGRAPH acfa2e0 DNW- 322 4693 --
+ attr: name (v/c), airport (v/c), city (v/c), state (v/c), country (v/c), lat (v/n),
| lon (v/n), weight (e/n)
+ edges from acfa2e0 (vertex names):
 [1] ABE->ATL ABE->DTW ABE->ORD ABI->DFW ABQ->ATL ABQ->BWI ABQ->CLT ABQ->DAL ABQ->DEN ABQ->DFW
[11] ABQ->HOU ABQ->IAH ABQ->JFK ABQ->LAS ABQ->LAX ABQ->MCI ABQ->MCO ABQ->MDW ABQ->MSP ABQ->OAK
[21] ABQ->ORD ABQ->PDX ABQ->PHX ABQ->SAN ABQ->SEA ABQ->SFO ABQ->SLC ABR->MSP ABY->ATL ACK->BOS
[31] ACK->DCA ACK->JFK ACT->DFW ACV->SFO ACY->ATL ACY->BOS ACY->DTW ACY->FLL ACY->MCO ACY->MYR
[41] ACY->ORD ACY->PBI ACY->RSW ACY->TPA ADK->ANC ADQ->ANC AEX->ATL AEX->DFW AEX->IAH AGS->ATL
[51] AGS->LGA AKN->ANC ALB->ATL ALB->BWI ALB->CLT ALB->DEN ALB->DTW ALB->EWR ALB->FLL ALB->IAD
[61] ALB->LAS ALB->MCO ALB->MDW ALB->MSP ALB->ORD ALB->RSW ALB->TPA ALO->ORD AMA->DAL AMA->DEN
+ ... omitted several edges
```

```
> as_adj(as.igraph(tgra))
322 x 322 sparse Matrix of class "dgCMatrix"
   [[ suppressing 43 column names ‘ABE’, ‘ABI’, ‘ABQ’ ... ]]
   [[ suppressing 43 column names ‘ABE’, ‘ABI’, ‘ABQ’ ... ]]

ABE . . . . . 1 . . . . . .....
ABI . . . . . .....
ABQ . . . . . 1 . . . . . .....
ABR . . . . . .....
ABY . . . . . 1 . . . . . .....
```

Network Analysis

Reading
- Og2016 Section 6. Network and node descriptives
- Og2016 Section 7. Distances and paths

Topics to cover

- Graph level measures
  - Density
  - Triangles and transitivity (or clustering coefficient)
  - Connected components
- Node level measures
  - Degree (in-degree / out-degree for directed graph)
  - Centrality (closeness, betweenness, eigenvector, PageRank)

Density

- The density of a network is the ratio of the number of “realized” edges to the all possible number of edges.

$$d = \frac{2 \cdot |E|}{|V| \cdot (|V| - 1)} \qquad \text{for undirected graph}$$

$$d = \frac{|E|}{|V| \cdot (|V| - 1)} \qquad \text{for directed graph}$$

```
1  edge_density(grr)
2  edge_density(grf)
```

```
> edge_density(grr)
[1] 0.2222222
> edge_density(grf)
[1] 0.003968254
```

Triangles and transitivity

- When there are more than three nodes, social network analysis gets meaningful. In that sense, triangle is the building block of social network analysis.
- Transitivity (or clustering coefficient) is a measure of how clique a network is. Transitivity is defined as if x --- y and y --- z, then x --- z.

```
count_triangles(grt)
sum(count_triangles(grt))
length(triangles(grt))
transitivity(grt)
count_triangles(grr)
transitivity(grr)
```

```
> count_triangles(grt)
[1] 1 1 0 1 0
> sum(count_triangles(grt))
[1] 3
> length(triangles(grt))
[1] 3
> transitivity(grt)
[1] 0.4285714
> count_triangles(grr)
 [1] 0 0 0 0 0 0 0 0 0 0
> transitivity(grr)
[1] 0
```

Connected components

- In real world social networks, it is often the case where not all nodes are reachable from each other.
- The disjoint sets of nodes are called components. Nodes in each component are connected and nodes in different components are not connected.

```
1  count_components(grr)
2  count_components(grt)
3  count_components(graph_from_literal(A-B,C-D))
```

```
> count_components(grr)
[1] 1
> count_components(grt)
[1] 1
> count_components(graph_from_literal(A-B,C-D))
[1] 2
```

Degree

- Degree is the most basic and fundamental measures in network analysis.
- Degree is basically the number of nodes that are connected to the focal node.
- For directed graph, in-degree and out-degree are used.

```
1  V(grr)$degree <- degree(grr)
2  tgrr <- as_tbl_graph(grr)
3  tgrr %>% mutate(degree=centrality_degree())
4  tgrf %>%
5    mutate(indegree=centrality_degree(mode="in"),
   outdegree=centrality_degree(mode="out")) %>%
6    arrange(-outdegree)
```

```
> V(grr)$degree
 [1] 2 2 2 2 2 2 2 2 2 2
```

Centrality

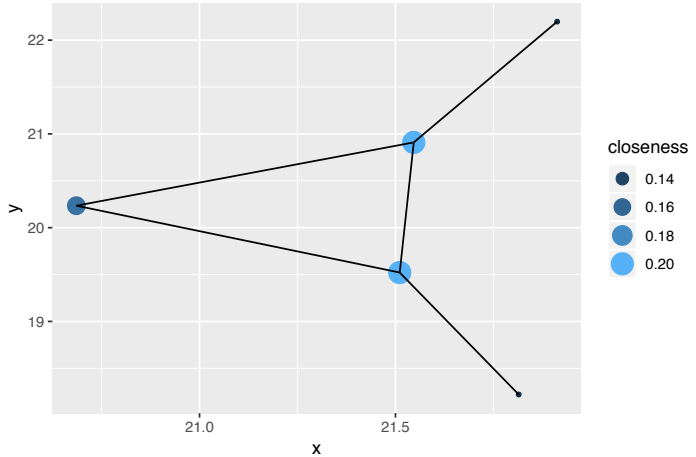
- One natural and key question in any network analysis is: “Which node is important?”
- The definition of importance depends on the context and the question.
- Degree is one form of centrality.
- Many centrality measures have been developed to capture different types of importance or prominence of nodes in a network.
- Among them are:
  - Closeness centrality
  - Betweenness centrality
  - Eigenvector centrality
  - PageRank
- Reading: https://en.wikipedia.org/wiki/Centrality

Closeness centrality

- Nodes that are on average close to other nodes get a high score.

```
1  closeness(grr)
2  closeness(grt)
3  as_tbl_graph(grt) %>%
4    mutate(closeness=centrality_closeness()) %>% ggraph() +
5    geom_edge_link() +
6    geom_node_point(aes(size=closeness, color=closeness)) +
7    scale_color_continuous(guide = 'legend')
```

```
> closeness(grr)
 [1] 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04 0.04
> closeness(grt)
        A         B         C         D         E
0.1666667 0.2000000 0.1250000 0.2000000 0.1250000
```

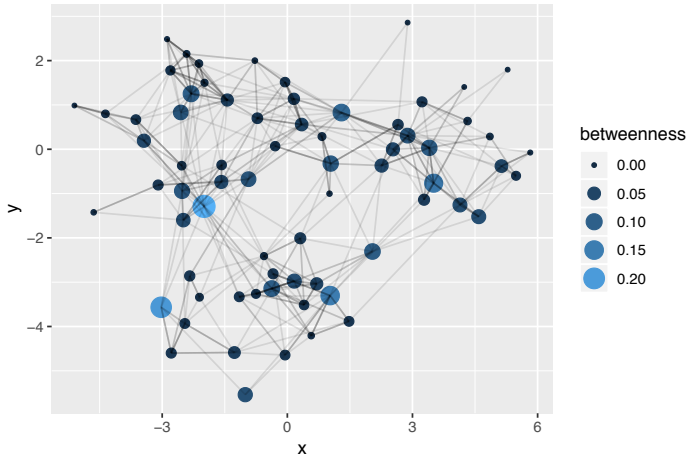


Betweenness centrality

- Nodes sitting in critical junctions score high.

```
1  betweenness(grr, normalized=T)
2  betweenness(grt)
3  as_tbl_graph(grh) %>%
4    mutate(betweenness=centrality_betweenness(normalized=T)) %>% ggraph() +
5    geom_edge_link(alpha=.1) +
6    geom_node_point(aes(size=betweenness, color=betweenness)) +
7    scale_color_continuous(guide = 'legend')
```

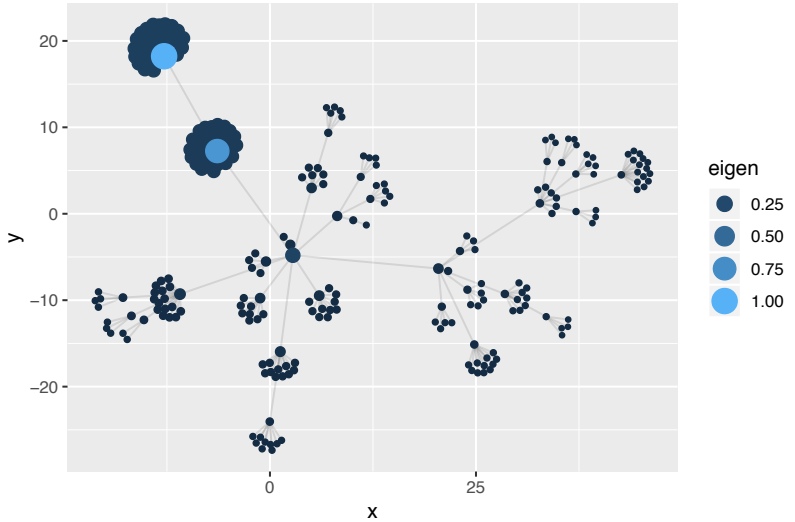
```
> betweenness(grr, normalized=T)
 [1] 0.2222222 0.2222222 0.2222222 0.2222222 0.2222222 0.2222222 0.2222222 0.2222222 0.2222222
[10] 0.2222222
> betweenness(grt)
A B C D E
0 3 0 3 0
```



Eigenvector centrality

- Nodes connected to highly influential nodes get a high score.

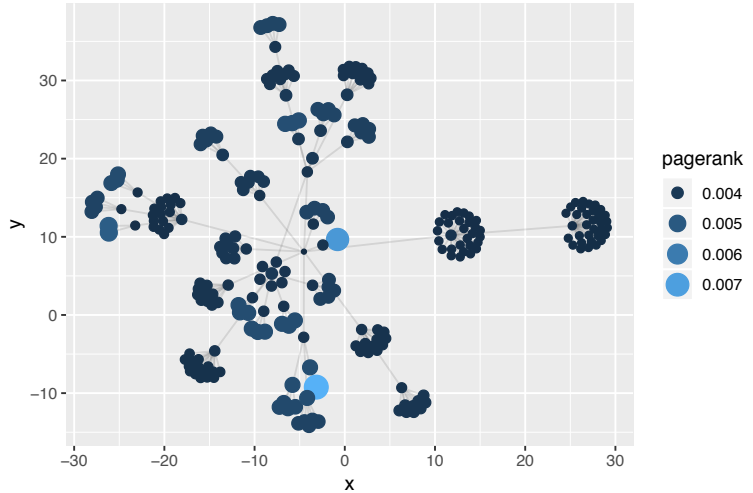
```
as_tbl_graph(grf) %>%
  mutate(eigen=centrality_eigen()) %>% ggraph() +
  geom_edge_link(alpha=.1) +
  geom_node_point(aes(size=eigen, color=eigen)) +
  scale_color_continuous(guide = 'legend')
```



PageRank

- PageRank is a variant of eigenvector centrality.
  Reading: https://cambridge-intelligence.com/eigencentrality-pagerank/

```
as_tbl_graph(grf) %>%
  mutate(pagerank=centrality_pagerank()) %>% ggraph() +
  geom_edge_link(alpha=.1) +
  geom_node_point(aes(size=pagerank, color=pagerank)) +
  scale_color_continuous(guide = 'legend')
```



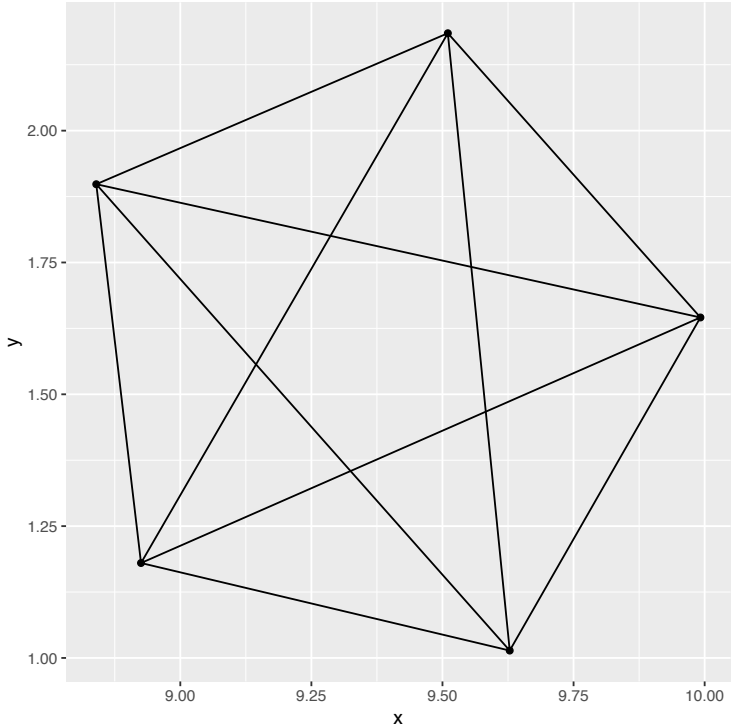
Network Topology

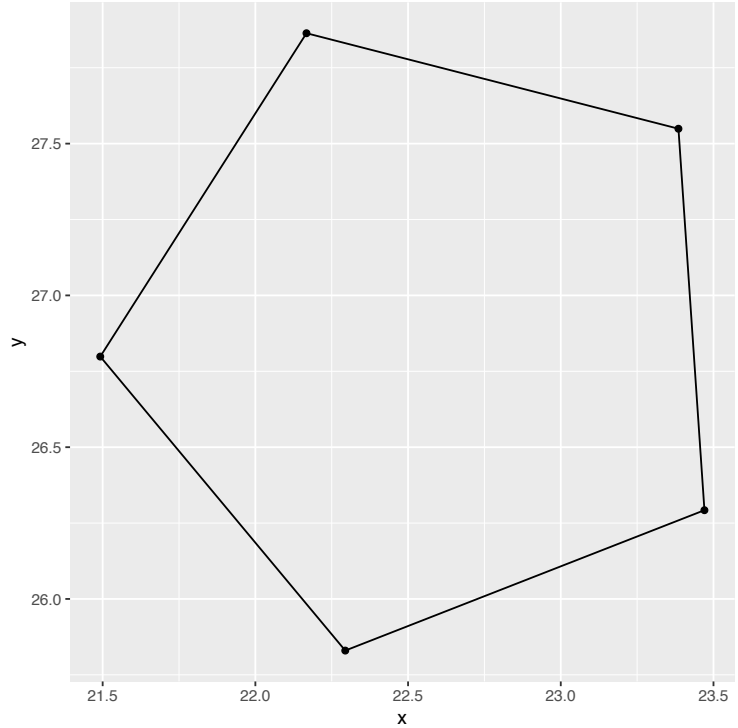
Archetypal topologies

- Complete graph
- Barbell or lollipop graph
- Cycle or ring graph
- Star graph or tree
- Wheel graph
- Path graph
- Ladder graph or lattice

Complete graph and ring graph

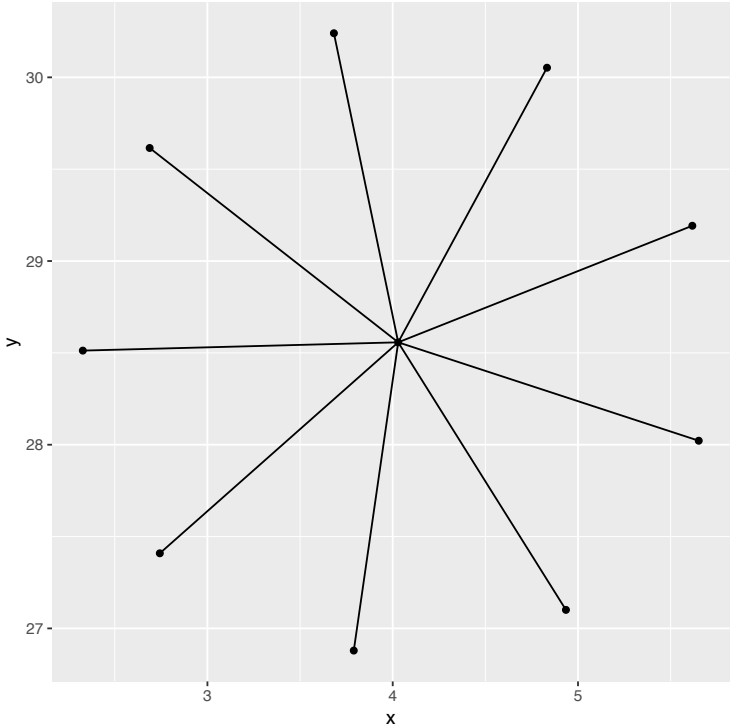
```
1  ggraph(make_full_graph(5)) + geom_edge_link() + geom_node_point()
2  ggraph(make_ring(5)) + geom_edge_link() + geom_node_point()
```

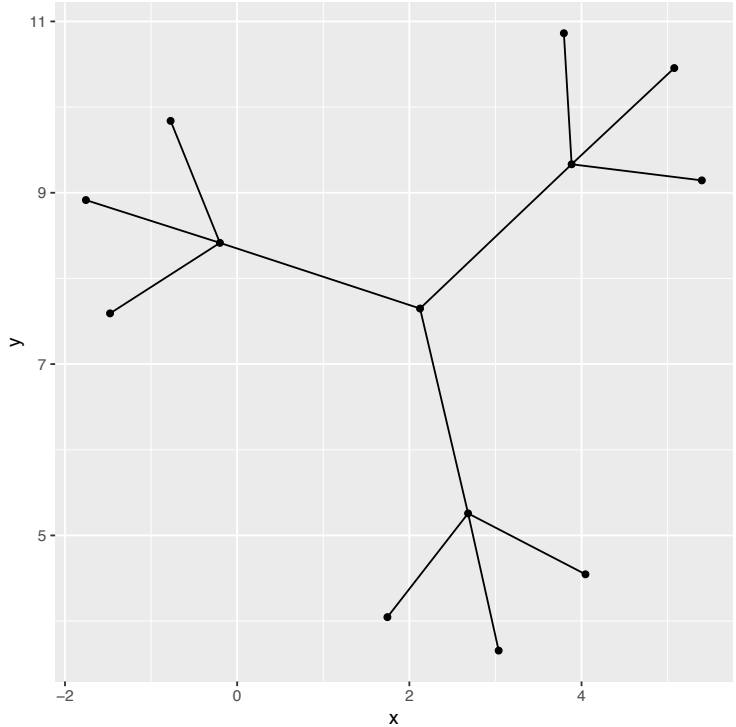




Star graph and tree

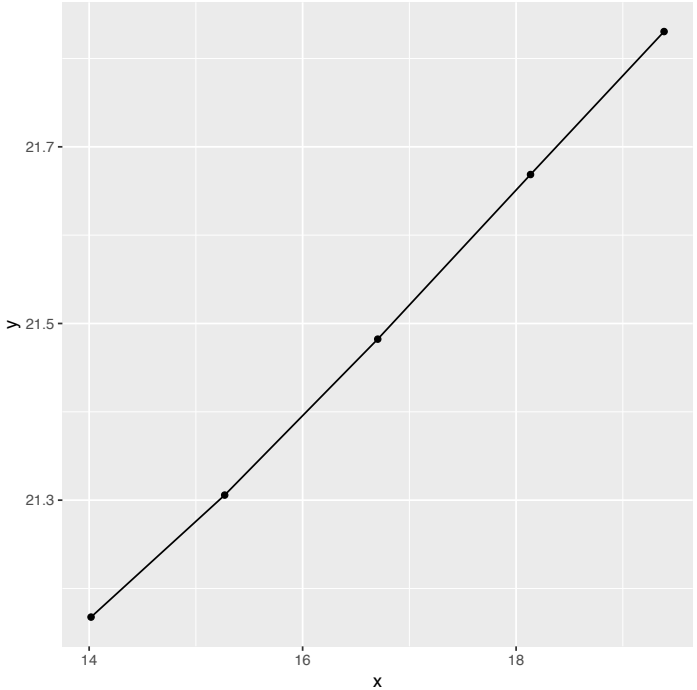
```
1  ggraph(make_star(10)) + geom_edge_link() + geom_node_point()
2  ggraph(make_tree(1+3+3^2, children=3)) + geom_edge_link() + geom_node_point()
```

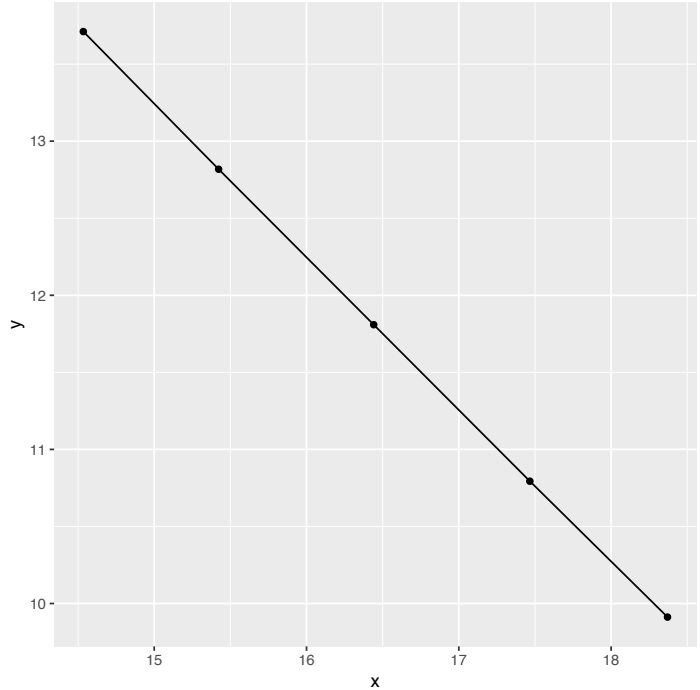




Path graph

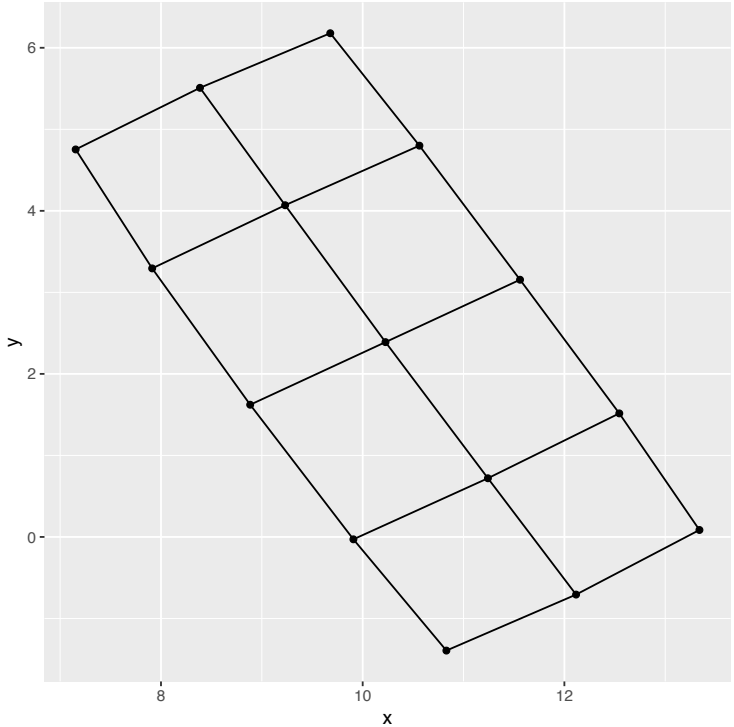
```
1  ggraph(make_empty_graph(n=5)+path(1,2,3,4,5)) + geom_edge_link() +
   geom_node_point()
2  ggraph(make_lattice(c(5))) + geom_edge_link() + geom_node_point()
```

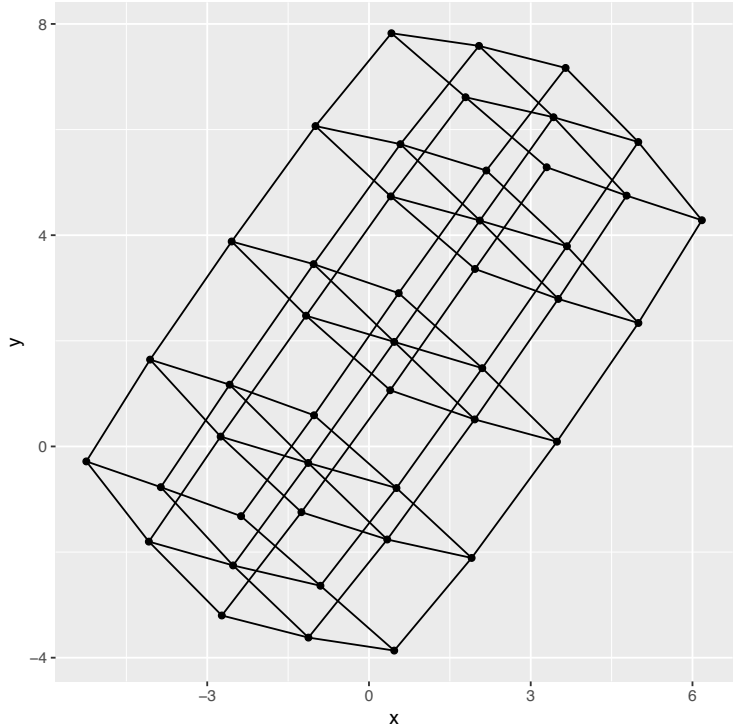




Lattice

```
1  ggraph(make_lattice(c(5,3))) + geom_edge_link() + geom_node_point()
2  ggraph(make_lattice(c(5,3,3))) + geom_edge_link() + geom_node_point()
```





Network types

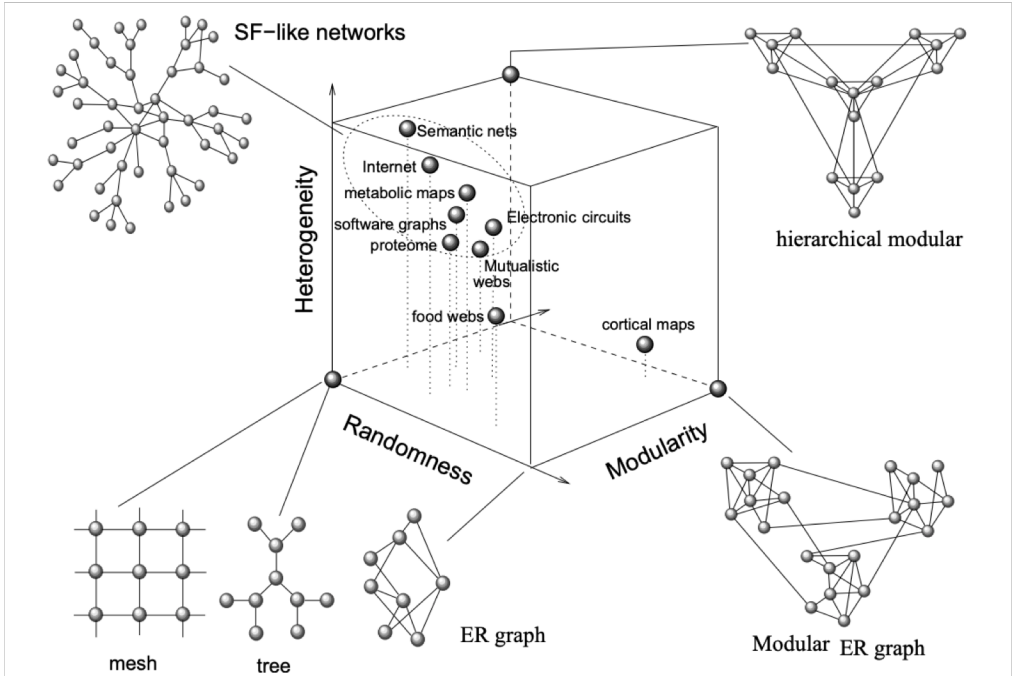


FIG. 3 A zoo of complex networks. In this qualitative space, three relevant characteristics are included: randomness, heterogeneity and modularity. The first introduces the amount of randomness involved in the process of network's building. The second measures how diverse is the link distribution and the third would measure how modular is the architecture. The position of different examples are only a visual guide. The domain of highly heterogeneous, random hierarchical networks appears much more occupied than others. Scale-free like networks belong to this domain.

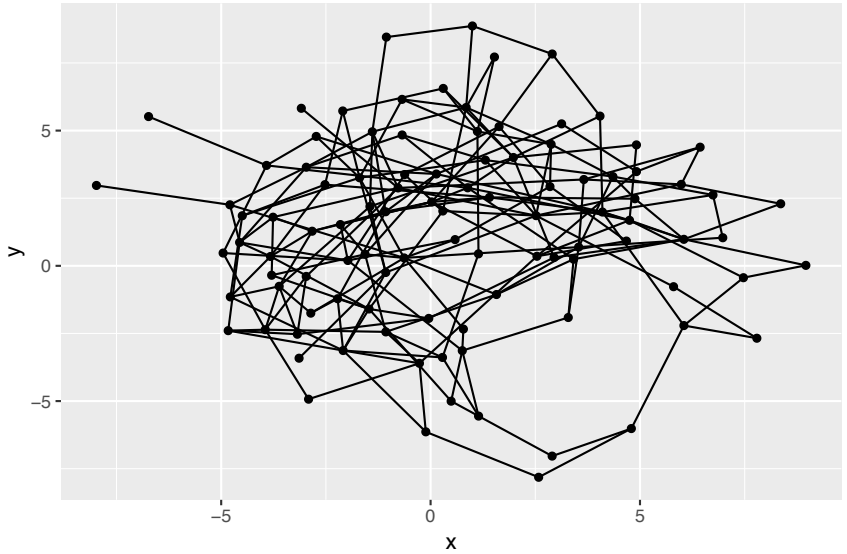
Solé R.V., Valverde S. (2004) Information Theory of Complex Networks: On Evolution and Architectural Constraints. In: Ben-Naim E., Frauenfelder H., Toroczkai Z. (eds) Complex Networks. Lecture Notes in Physics, vol 650. Springer, Berlin, Heidelberg

Figure from http://complex.upf.es/~ricard/INFONETS.pdf

Random network

- Erdos-Renyi model is the simplest yet fundamental random graph model.
- For a given number of nodes, all edges are realized with the same probability.

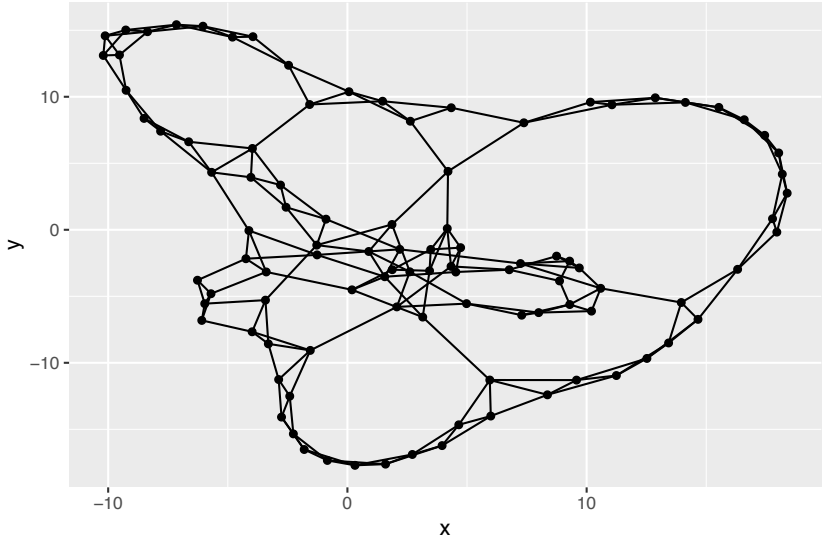
```
1  gr_er <- play_erdos_renyi(100,m=200)
2  ggraph(gr_er) + geom_node_point() + geom_edge_link()
```



Small-world network

- Watts-Strogatz model generates a network with the small-world property.
- The small-world property includes short average path length, high clustering.

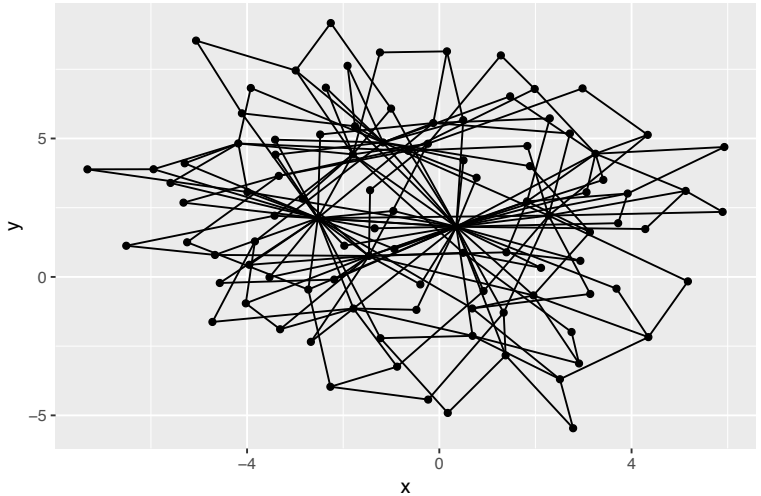
```
gr_sw <- play_smallworld(1, 100, 2, 0.05)
ggraph(gr_sw) + geom_node_point() + geom_edge_link()
```



Scale-free network

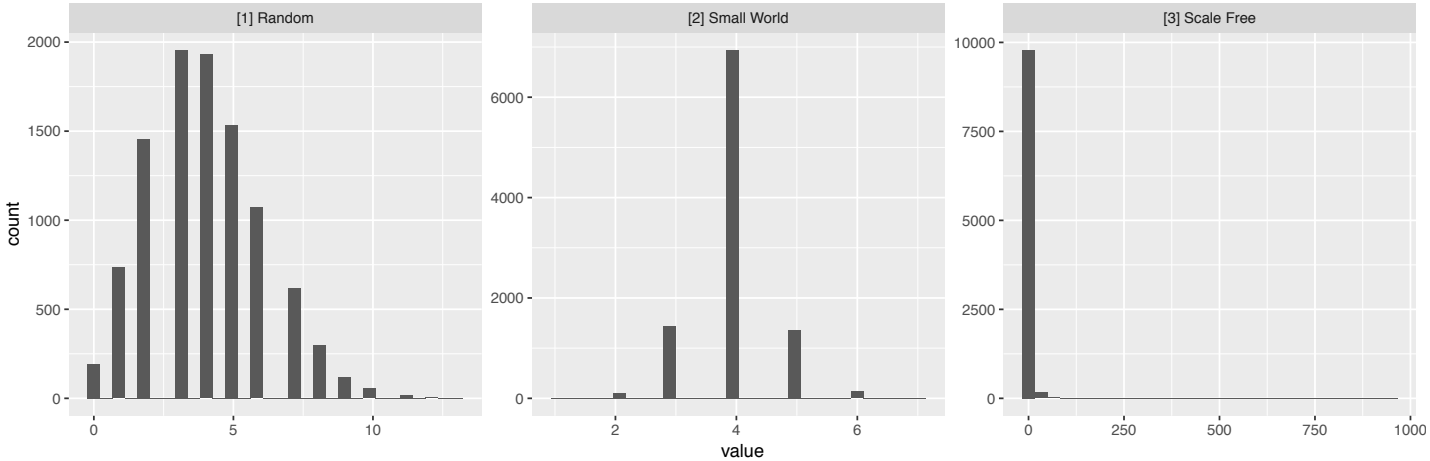
- Barabasi-Albert model creates a scale-free network with preferential attachment mechanism.
- Reading: https://www.scientificamerican.com/article/scale-free-networks/

```
gr_sf <- play_barabasi_albert(100, 1, 2)
ggraph(gr_sf) + geom_node_point() + geom_edge_link()
```



Degree distribution

```
1  gr_er_10k <- play_erdos_renyi(10000,m=20000)
2  gr_sw_10k <- play_smallworld(1, 10000, 2, 0.05)
3  gr_sf_10k <- play_barabasi_albert(10000, 1, 2)
4  dder <- as_tibble(degree(gr_er_10k)) %>% mutate(type="[1] Random")
5  ddsw <- as_tibble(degree(gr_sw_10k)) %>% mutate(type="[2] Small World")
6  ddsf <- as_tibble(degree(gr_sf_10k)) %>% mutate(type="[3] Scale Free")
7  ggplot(bind_rows(dder, ddsw, ddsf)) +
8    geom_histogram(aes(x=value)) +
9    facet_wrap(~type, scale="free")
```



Interesting characteristics of real-world networks

- Both small world network and scale-free network capture and characterize certain aspects of real-world networks.
- Short average path length
  - Readings
    - https://en.wikipedia.org/wiki/Six_degrees_of_separation
    - https://en.wikipedia.org/wiki/Small-world_experiment
- The strength of the weak ties (Granovetter, 1973)
- Disproportionate degree distribution (The Matthew effect)
  - One of the mechanism to generate such a severely right-skewed distribution is preferential attachment.
  - As a new node enters the network, an existing node with a high degree gets higher probability to get connected the new node.

Network Visualization

Reading

- Og2018 Section 1. Introduction: network visualization
- Og2018 Section 4. Plotting networks with igraph
- Blog posts on ggraph: Introduction, Layouts, Nodes, Edges

Network visualization process

- Step 1: Define what relationships between what entities you want to analyze.
- Step 2: Construct a network object and compute measures of interest.
- Step 3: Compute a layout of nodes.
- Step 4: Assign data to visual elements such as size, color, thickness, alpha, etc.
- Step 5: Visualize and iterate Steps 2 through 5.

Tutorials

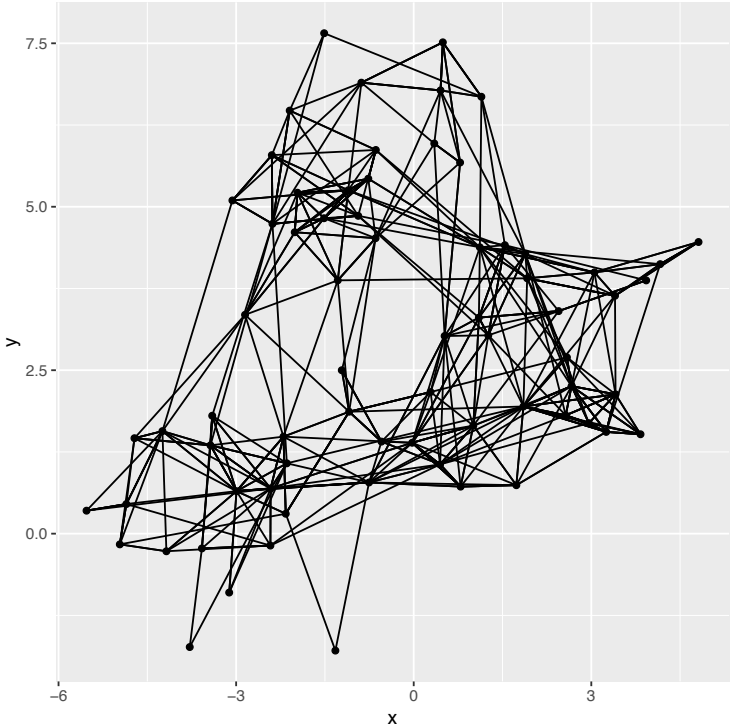
- Like you took Data Camp courses in other modules, read the following tutorials and make sure you can replicate the graphical artifacts by yourself.
- Blog posts by Thomas Lin Pederson
  - Announcing ggraph: A grammar of graphics for relational data
  - Introduction to ggraph: Layouts
  - Introduction to ggraph: Nodes
  - Introduction to ggraph: Edges

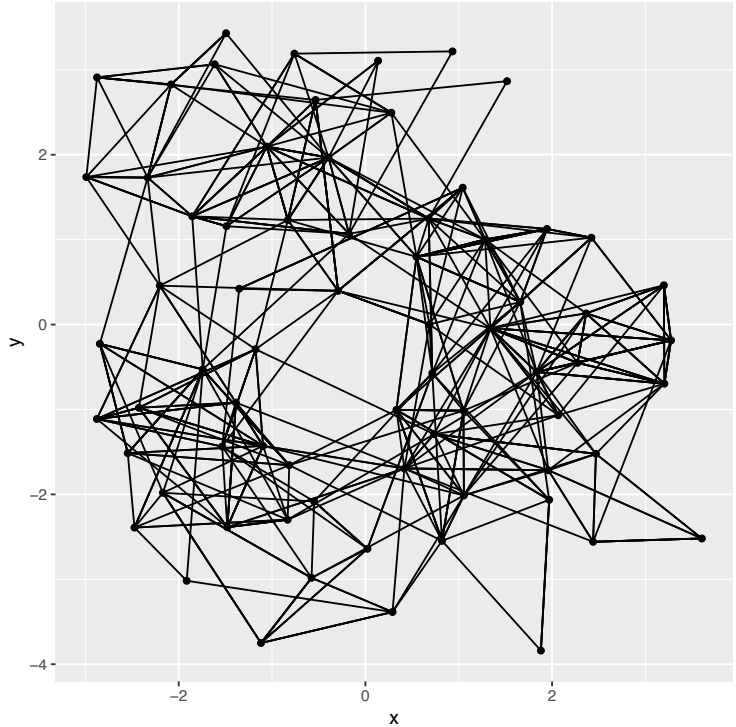
Layouts

- Force-directed layout
- Linear and circular layouts
- Treemap and circlepack layouts
- Partition and dendrogram layouts

Force-directed layout

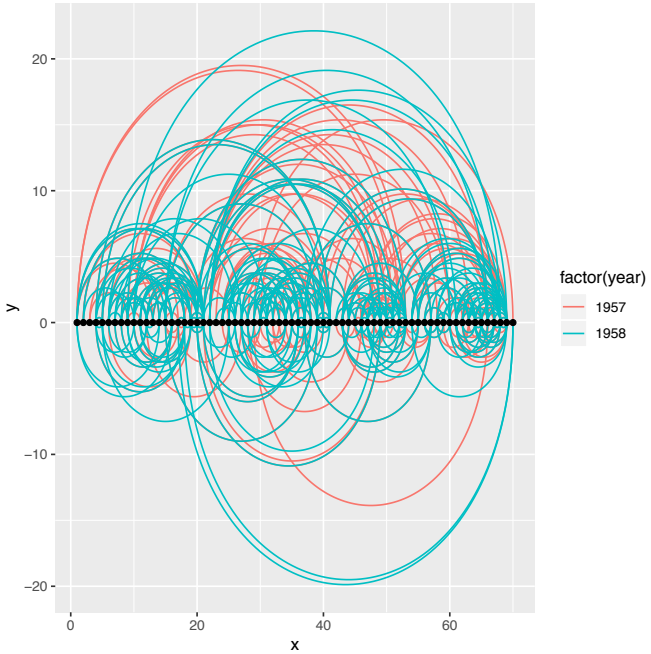
```
1  ggraph(grh, "nicely") + geom_edge_link() + geom_node_point()
2  ggraph(grh, "kk") + geom_edge_link() + geom_node_point()
```

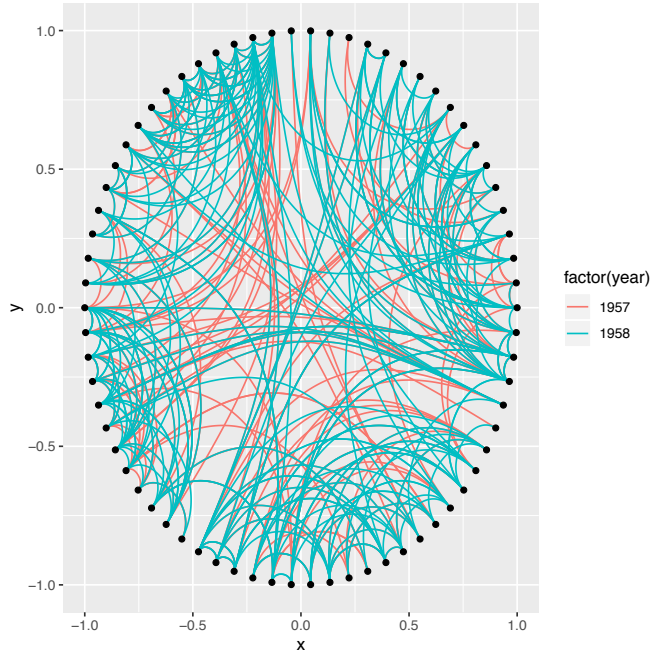




Linear and circular layouts

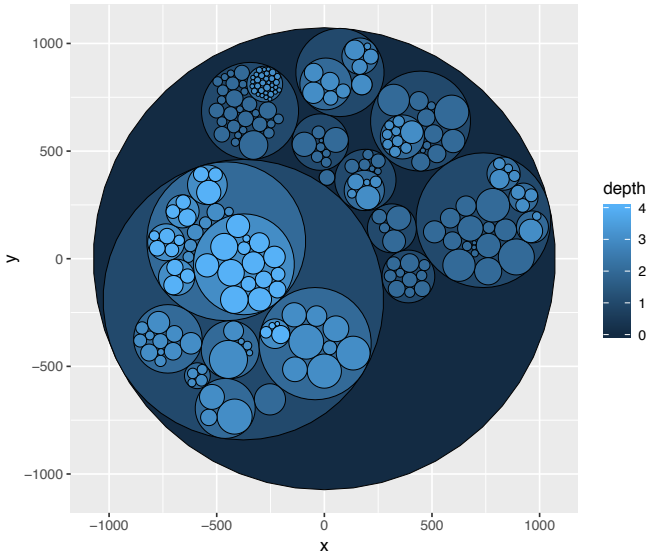
```
1  ggraph(grh, "linear") + geom_edge_arc(aes(color=factor(year))) +
   geom_node_point()
2  ggraph(grh, "linear", circular=T) + geom_edge_arc(aes(color=factor(year))) +
   geom_node_point()
```

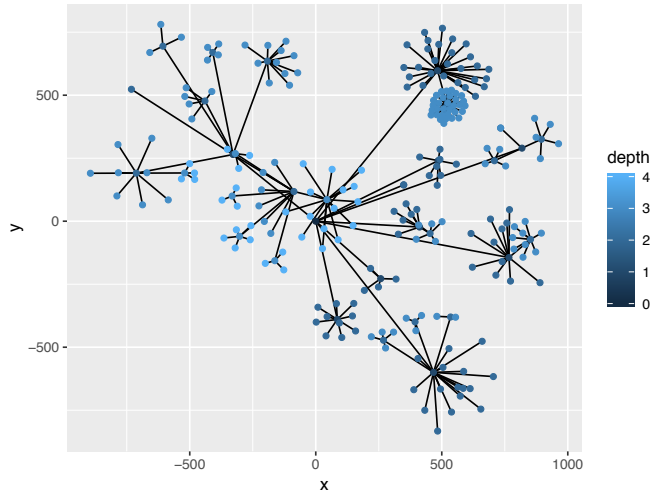




Circlepack layout

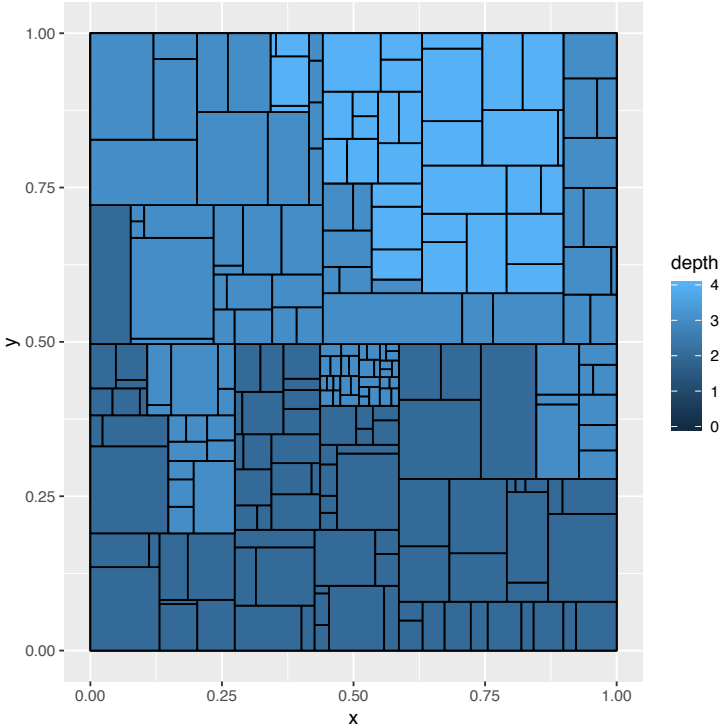
```
1  ggraph(grf, "circlepack", weight="size") + geom_node_circle(aes(fill =
   depth), size = 0.25, n = 50) + coord_fixed()
2  ggraph(grf, "circlepack", weight="size") + geom_edge_link() +
   geom_node_point(aes(color=depth)) + coord_fixed()
```

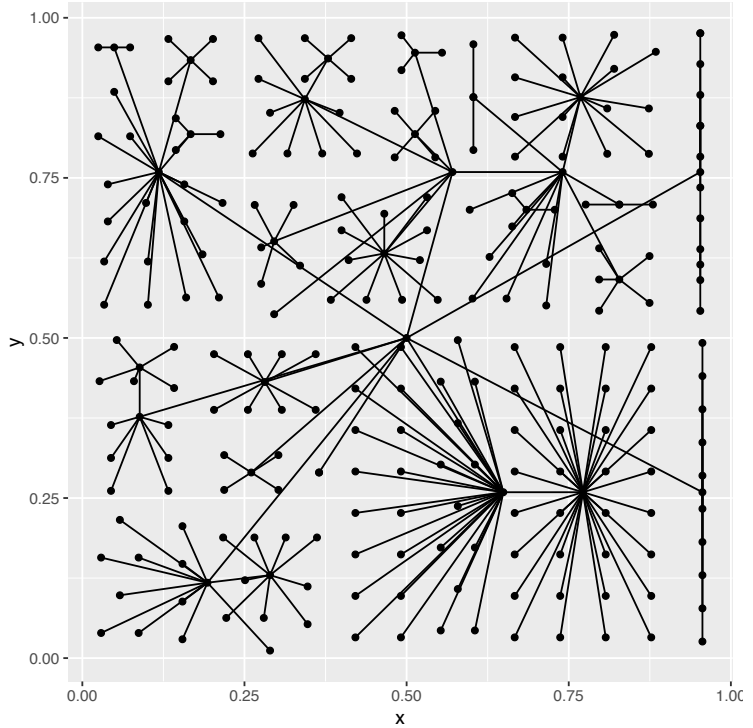




Treemap layout

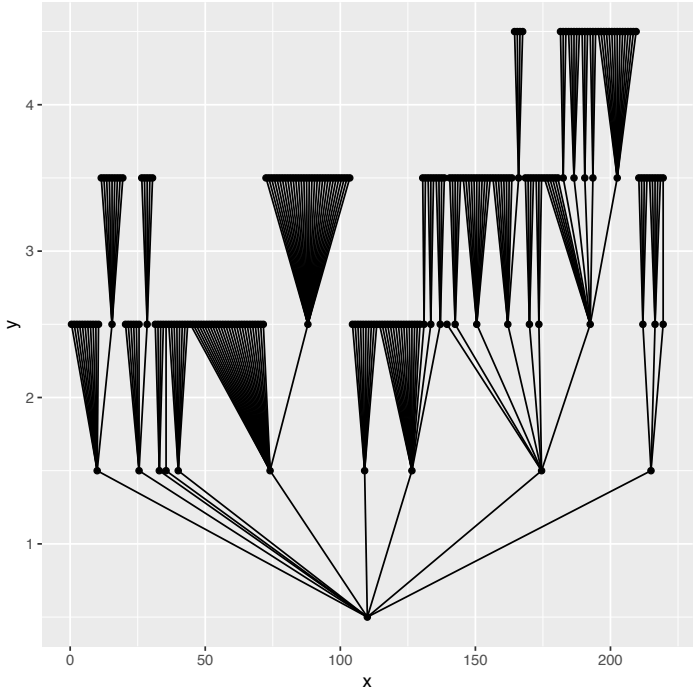
```
1  ggraph(grf, "treemap", weight="size") + geom_node_tile(aes(fill=depth))
2  ggraph(grf, "treemap") + geom_edge_link() + geom_node_point()
```

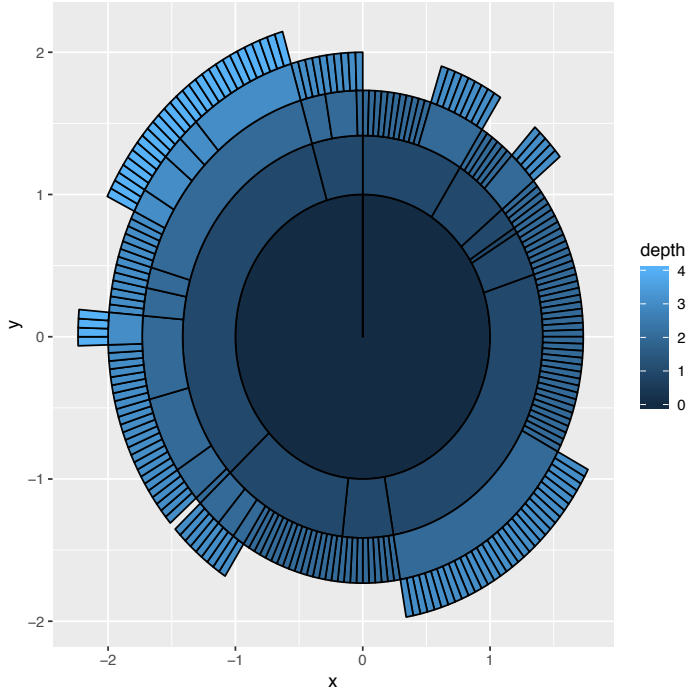




Partition layout

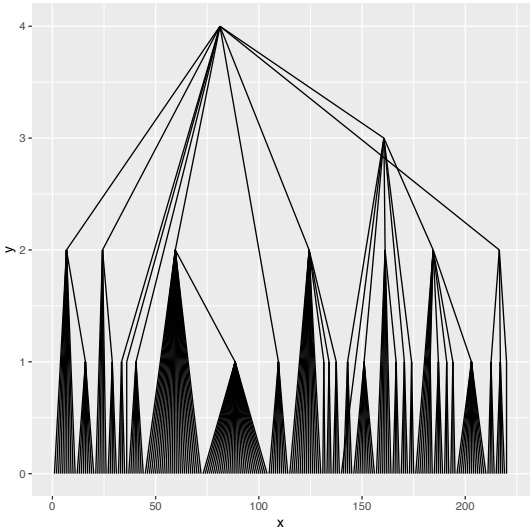
```
1  ggraph(grf, "partition") + geom_edge_link() + geom_node_point()
2  ggraph(grf, "partition", circular=T) + geom_edge_link() +
   geom_node_arc_bar(aes(fill=depth))
```

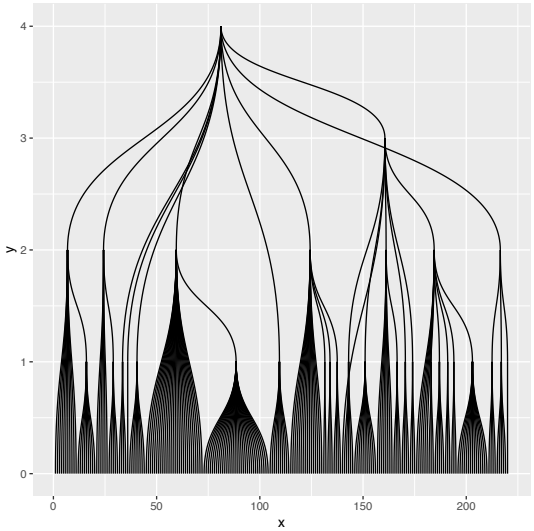


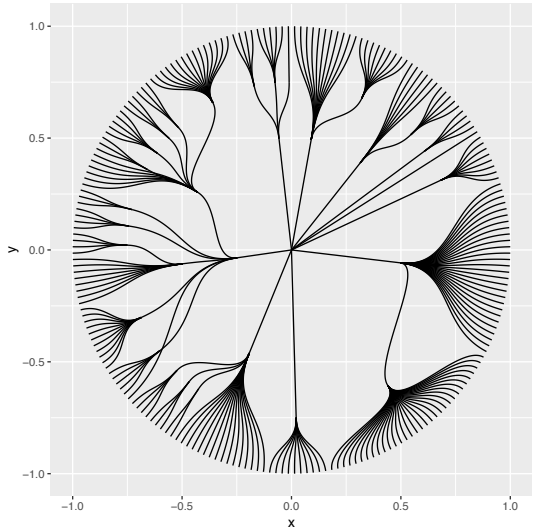


Dendrogram layout

```
1  ggraph(grf, "dendrogram") + geom_edge_link()
2  ggraph(grf, "dendrogram") + geom_edge_diagonal()
3  ggraph(grf, "dendrogram", circular=T) + geom_edge_diagonal()
```







Weekly Recap

Things we covered this week

- Anatomy of Networks
  - Terminology
  - Handling network objects with igraph and tidygraph
- Network Analysis
  - Density
  - Transitivity (or clustering)
  - Components
  - Degree
  - Centrality
- Network Topology
  - Archetypal topologies
  - Random network
  - Small-world network
  - Scale-free network
- Network Visualization
  - Layout
  - Node geoms
  - Edge geoms

Things to do this week

- 4 Quizzes (Due: Thursday, February 7, 11:59pm)
  - Week 5.1. Anatomy of Networks
  - Week 5.2. Network Analysis
  - Week 5.3. Network Topology
  - Week 5.4. Network Visualization
- 1 Weekly Problem (Due: Friday, February 8, 11:59pm)
- 1 Bi-weekly Assignment (Due: Saturday, February 9, 11:59pm)