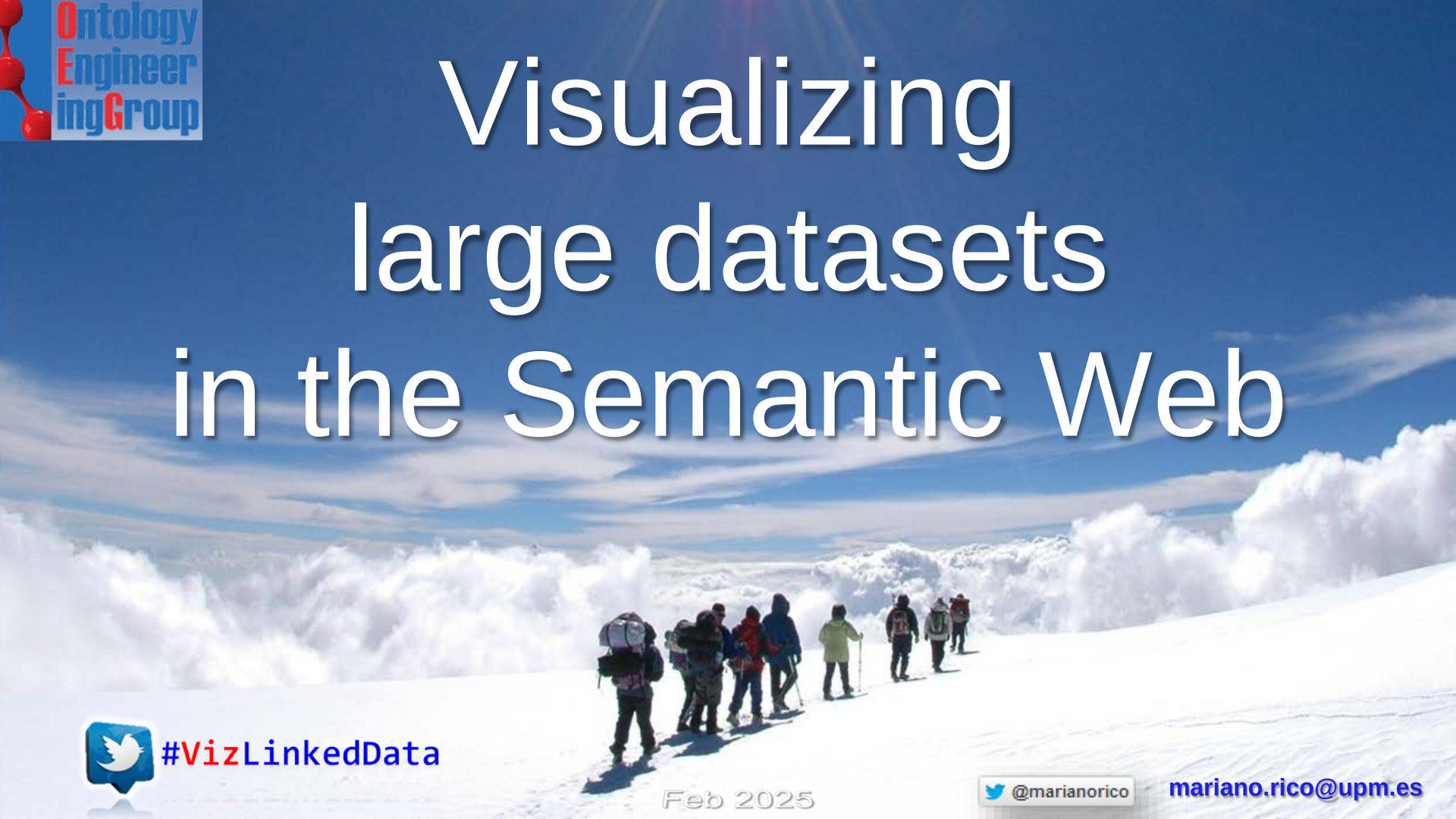
Visualizing large datasets in the Semantic Web



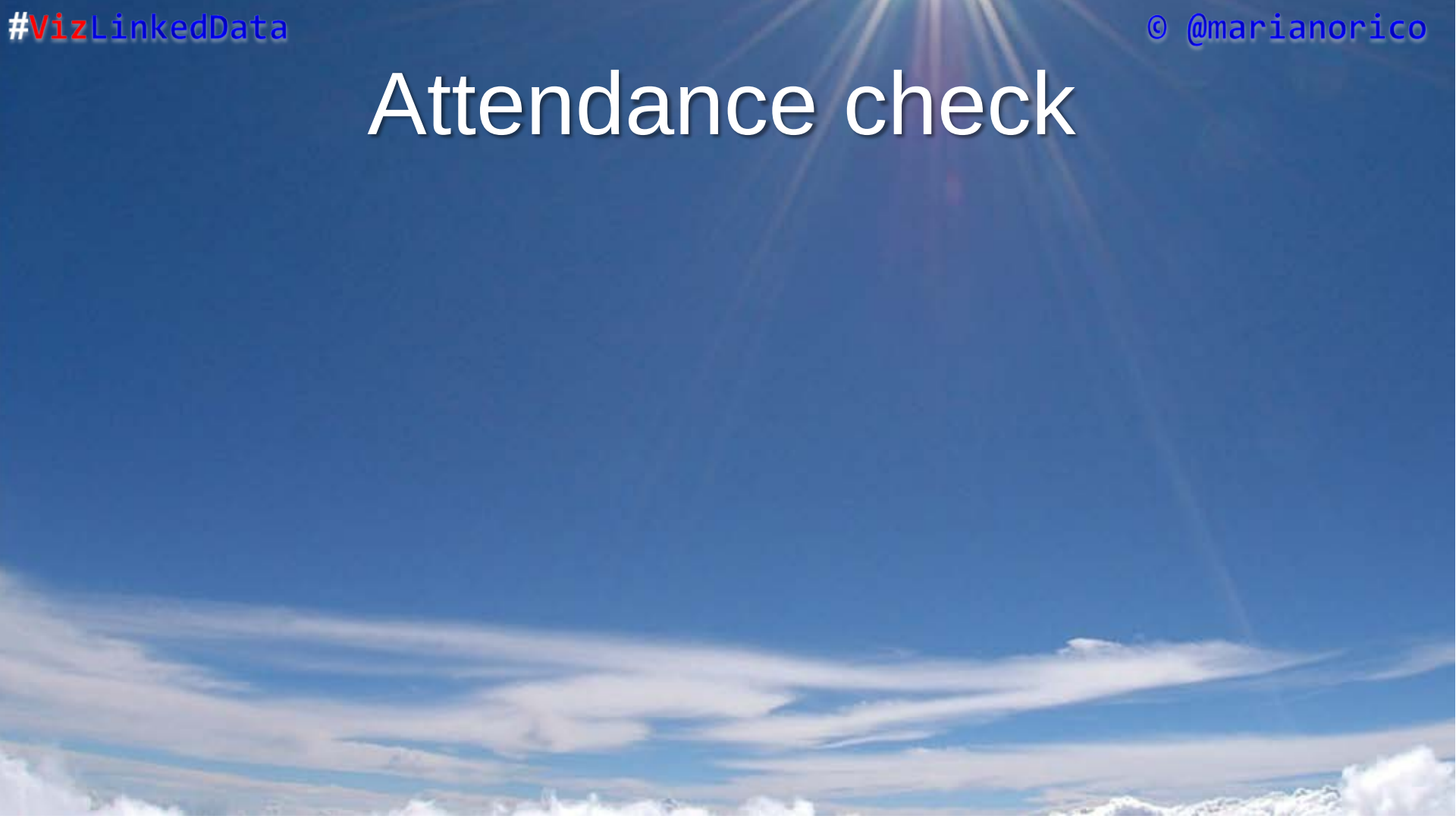
#VizLinkedData

Feb 2025

@marianorico

mariano.rico@upm.es

Attendance check



Get the material

- Slides and files can be downloaded from

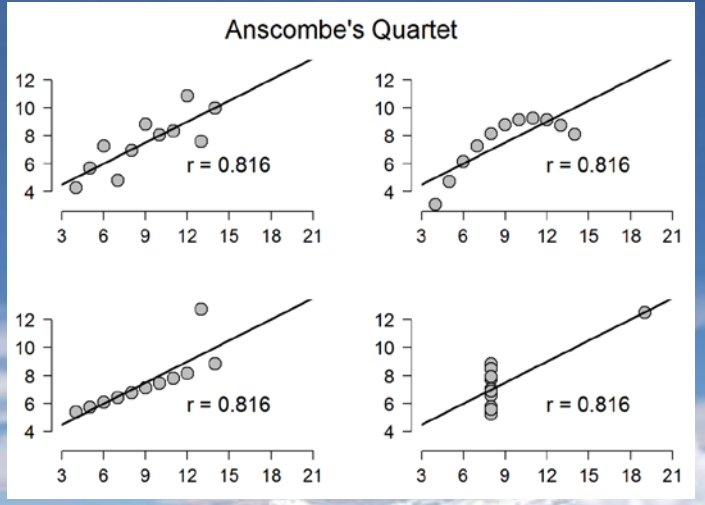
https://tinyurl.com/bigdatavizMRA

Why visualizing?

- Numerical analysis can miss info. A graph can get that info
  - Classical example: Anscombe's Quartet (1973)

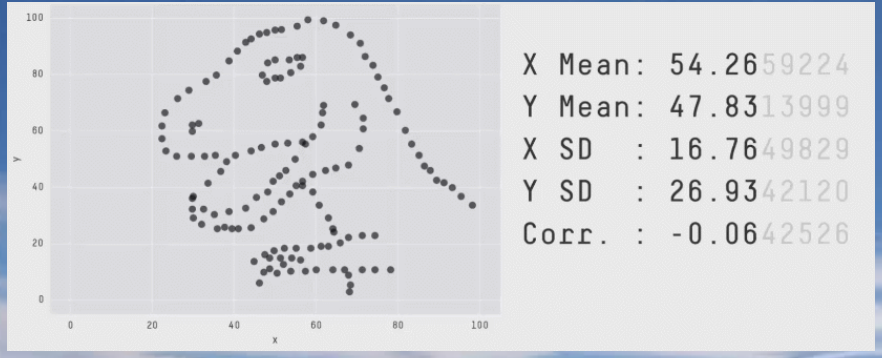
For all four datasets:

Property	Value	Accuracy
Mean of x	9	exact
Sample variance of x : s_x^2	11	exact
Mean of y	7.50	to 2 decimal places
Sample variance of y : s_y^2	4.125	±0.003
Correlation between x and y	0.816	to 3 decimal places
Linear regression line	$y = 3.00 + 0.500x$	to 2 and 3 decimal places, respectively
Coefficient of determination of the linear regression : R^2	0.67	to 2 decimal places



Why visualizing?

- Numerical analysis can miss info. A graph can get that info
  - Modern example: The Datasaurus Dozen (A. Cairo 2017)



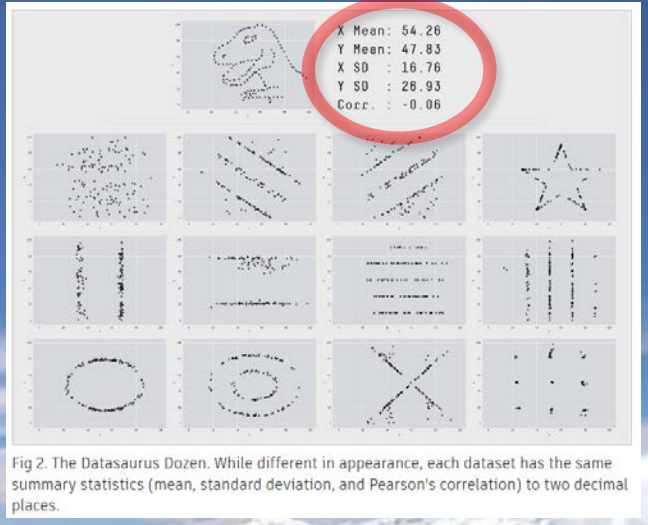


Fig 2. The Datasaurus Dozen. While different in appearance, each dataset has the same summary statistics (mean, standard deviation, and Pearson's correlation) to two decimal places.

Content

- Large-scale graph analytics
- Visualizing big graphs
- The LOD cloud as a big data use case
- Big graphs with Gephi. An intro
  - Gephi Layouts
  - Gephi + DBpedia
  - Gephi + Wikidata
  - Gephi + RDF/OWL
  - Gephi + R
- Igraph
- Other tools
  - Loupe. Explore datasets
  - RelFinder. A first graphical approach
  - Gruff. Graph SPARQL query results

Viewing **without** visualizing

LARGE-SCALE GRAPH ANALYTICS

Problem

- Huge graphs
  - Facebook: 1 billion users ➔ 1G nodes (aka vertex) graph
  - Google: the indexed web is a ~55 billion pages graph (Common Crawl has 25G as of April 2020)
  - Wayback Machine (Internet Archive) has ~654 billion pages (as of Feb. 2022)
- Complex/heavy algorithms
  - Page rank
  - Shortest path in a graph

State of the art

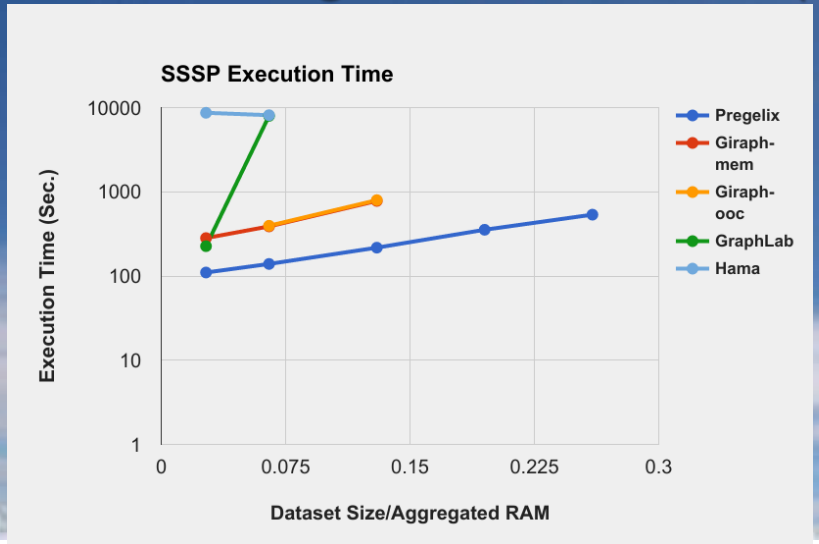
- Pregelix
  - Open source (Java)
  - Distributed graph processing system
    - Designed to handle resources efficiently (memory and disc)

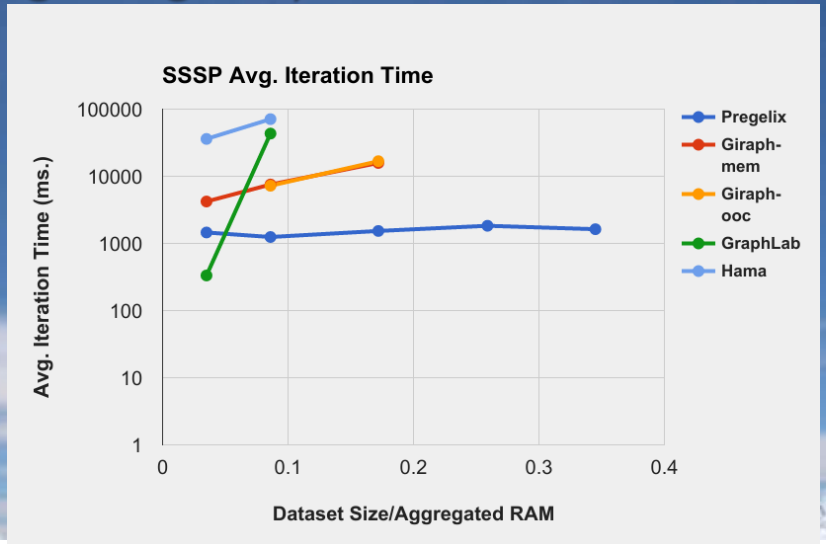
Pregelix

- Comparison with other Big Graph Analytics platforms (Apache Giraph, Apache GraphX, ~~GraphLab~~)
  - Specific problem: Single-Source Shortest Paths (SSSP) problem
    - Find the shortest paths from a source vertex to all other vertices in the graph.
      - Dijkstra's algorithm was the first successful algorithm
  - Configuration: 32 machine cluster

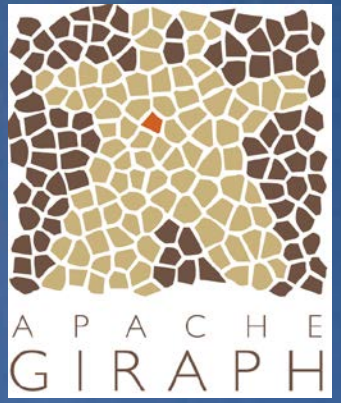
Pregelix

- Performance results (over a graph with 0.7B nodes and 6B edges)
  - end-to-end execution time (left figure)
  - average iteration time (right figure)





Apache Giraph



- Apache project (open source)
  - Java
- Exploits Apache hadoop
- Based on Google's Pregel
  - Pregel is C++
- Efficient. Used by Facebook. See here
- Based on the **BSP method**
  - BSP = Bulk Synchronous Parallel

BSP method

- Created in 80's by Leslie Valiant (Stanford U.)
  - See Leslie G. Valiant, "A bridging model for parallel computation", Communications of the ACM, Volume 33 Issue 8, Aug. 1990.
- Used by Pregel (Google) and Giraph (Apache)

BSP method

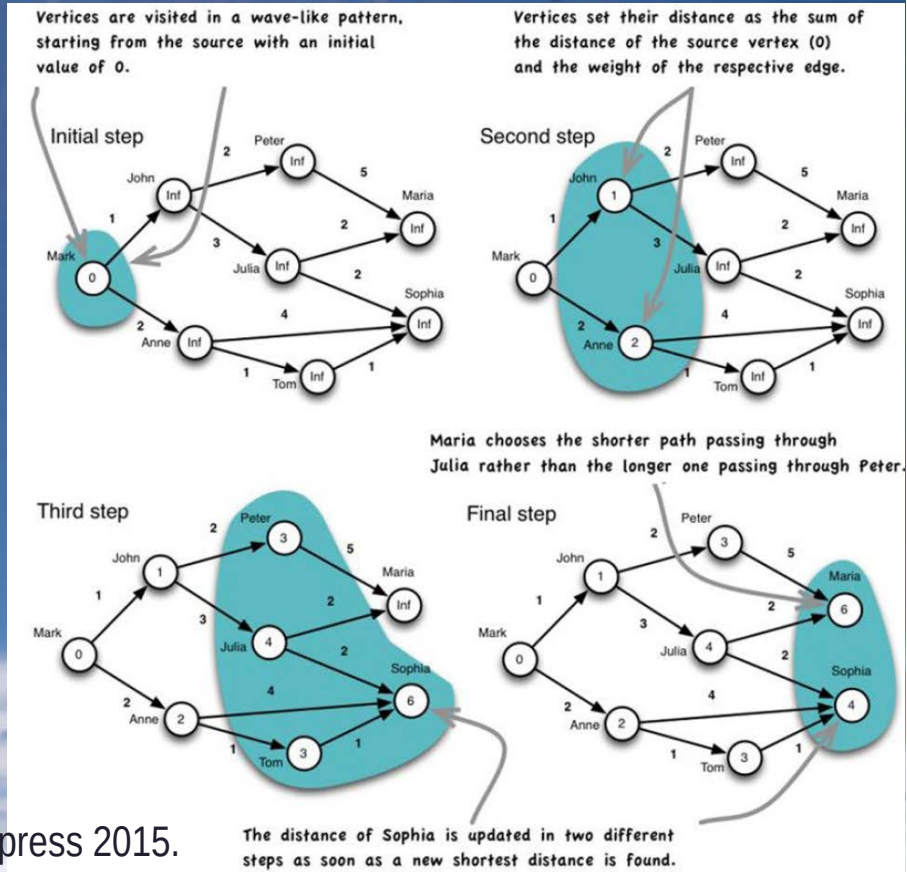
- The method
  - Algorithms as series of iterations (aka “supersteps”)
  - Vertex oriented. Each vertex in the graph:
    - Calls a method (in parallel)
    - Sends messages to the first-neighbors vertex
      - Will be read in the next superstep
    - Can read messages sent in the previous superstep
    - Can modify the state (create/delete) of the vertex edges

BSP method. An example

- Maximum age in a social network
  - Algorithm for BSP: In each iteration:
    - Each vertex sets its value to the maximum between its current value and the values of its (first) neighbors.
    - Check end of iterations: if none vertex updated its value, then finish iterations

BSP method. SSSP example

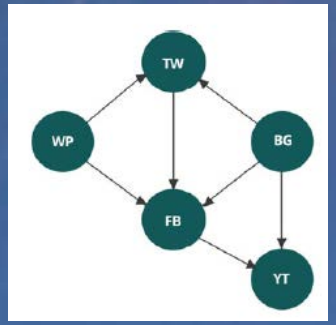
- SSSP algorithm for BSP.
  In each iteration:
  - Each vertex sets its distance value as the sum of its in-vertices + weight of the edge.
  - In case of several in-vertices, select the minimum
  - Check iterations end



Source: Practial Graph Analytics with Apache Giraph. Apress 2015.

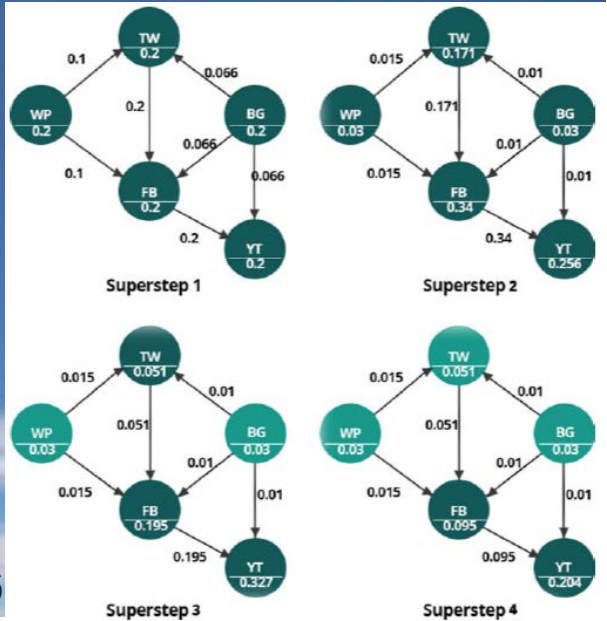
BSP method. PageRank example

- PageRank algorithm for BSP.
  - Initialization
    - Assign 0.2 (1/#vertices) to each vertex.
    - Message sent: val / #outEdges
  - In each iteration
    - Each vertex sums the incoming messages (im) * 0.85 and sums 0.15/#vertices (that is, computes pageRank value)
    - Check end of iterations



Initial state

Evolution



Source: Large scale graph porcessing using Apache Giraph. Springer 2016

BSP method. PageRank example

- Giraph Java code

Github source

```
public class PageRankVertexComputation extends
    BasicComputation<LongWritable, DoubleWritable, NullWritable, DoubleWritable> {

  public static final int MAX_SUPERSTEPS = 5; #ad hoc value for this example

  @Override
  public void compute(Vertex<LongWritable, DoubleWritable,
                              NullWritable> vertex,
                              Iterable<DoubleWritable> messages) throws IOException {
    if (getSuperstep() >= 1) {
      double sum = 0;
      for (DoubleWritable message : messages) {
        sum += message.get();
      }
     DoubleWritable vertexValue = new
                                  DoubleWritable( 0.15f / getTotalNumVertices() + 0.85f * sum);
      vertex.setValue(vertexValue);
    }
    if (getSuperstep() < MAX_SUPERSTEPS) {
      long edges = vertex.getNumEdges();
      sendMessageToAllEdges(vertex,
                                      new DoubleWritable(vertex.getValue().get() / edges));
    }
    else {
      vertex.voteToHalt();
    }
  }
}
```

Algorithms complexity

- Comparison by tool

Graph Theoretic Approaches for Analyzing Large-Scale Social Networks

Table 4. Algorithmic Time Complexities

FEATURES	NETWORKX	IGRAPH	GEPHI	PAJEK
Isomorphism	$O(N^2)$	exp	Na	Na
CORE M M=No. Of Lines	O(M)	O(M)	O(M)	O(M)
Cliques	O(|V|/(Log)2)	O(3|V|/3)	Na	O(N)
Shortest Path	O(|V|.|E|)	O(|V|+|E|)	O(|V|+|E|)	O(|V|+|E|)
Clustering	O(V)	Na	O(V)	Na
All Simple Path	O(|V|+|E|)	O(|V|+|E|)	Na	Na
Closeness Centrality	O(N.|E|)	O(N.|E|)	Na	Na
Density	$O(N^3)$	O(1)	Na	Na
MST	Na	O(|V|+|E|)	Na	Na
Cycles	O((|V|+|E|).C+1)	Na	Na	Na
PageRank	Na	O(E)	O(E)	Na
Betweenness	Na	O(|V|.|E|)	Na	Na
Eigenvector	Na	O(|V|+|E|)	Na	Na

Layouts

- Layouts supported comparison by tool

Table 3. Comparison Based on Graph Layout Supported by Tools

Layout	NETWORKX	IGRAPH	GEPHI	PAJEK
Circular Layout	Yes	Yes	Yes	Yes
Random Layout	Yes	Yes	Yes	No
Spectral Layout	Yes	No	No	No
Spring Layout	Yes	Yes	Yes	Yes
Graphviz Layout	Yes	No	No	No
Kamanda Kawai	No	Yes	Yes	No
Fruchterman Reingold	No	Yes	Yes	No
Force Atlas Layout	No	No	Yes	No

Graph Theoretic Approaches for Analyzing Large-Scale Social Networks

Finally!: viewing visualizing

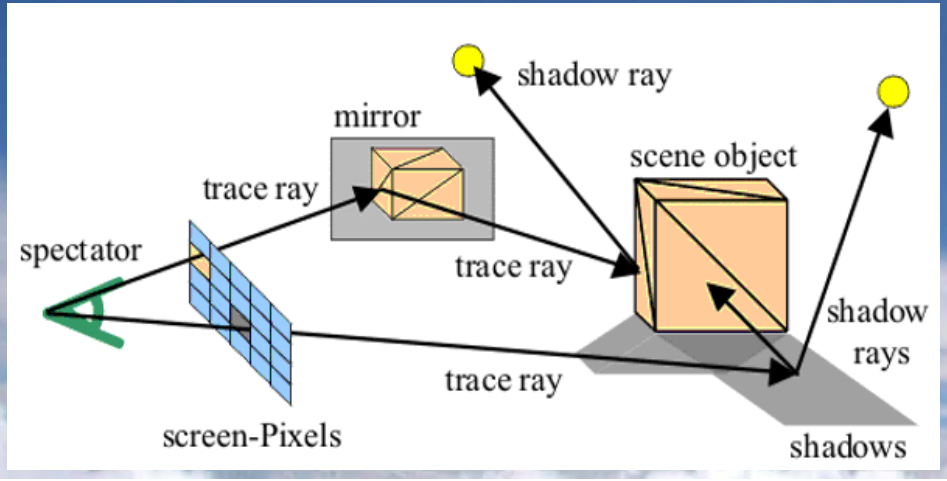
VISUALIZING BIG GRAPHS

Display limits

- Imagine a 100 billion star galaxy
- Can you see all the stars? Look for “Andromeda”
- Sure?
  - What is the screen resolution of your
    - Monitor? 4K = 8.3Mpixel in 16x9
    - Eyes? ~600Mpixel Source

Display limits

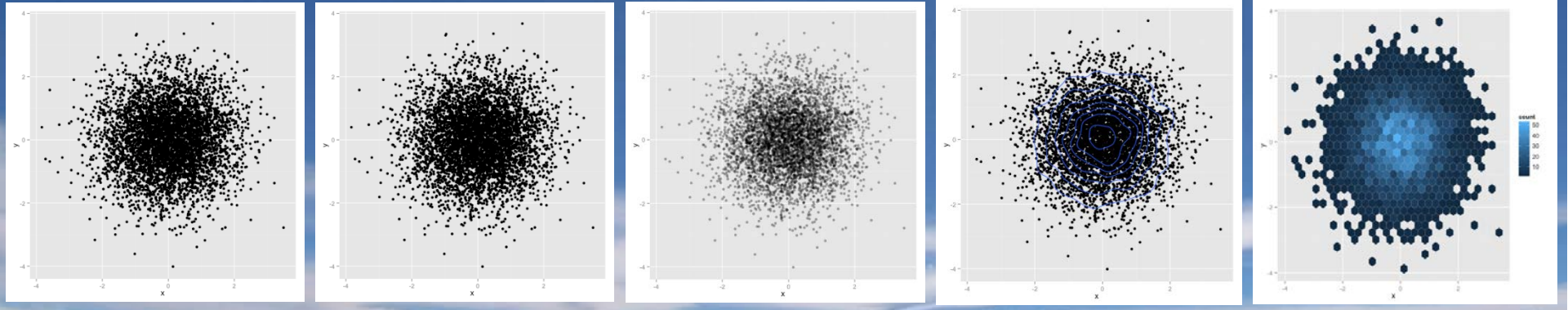
- It is like ray-tracing
  - Rendering is not a big problem
  - The problem is to make all the computation



Source

How to display

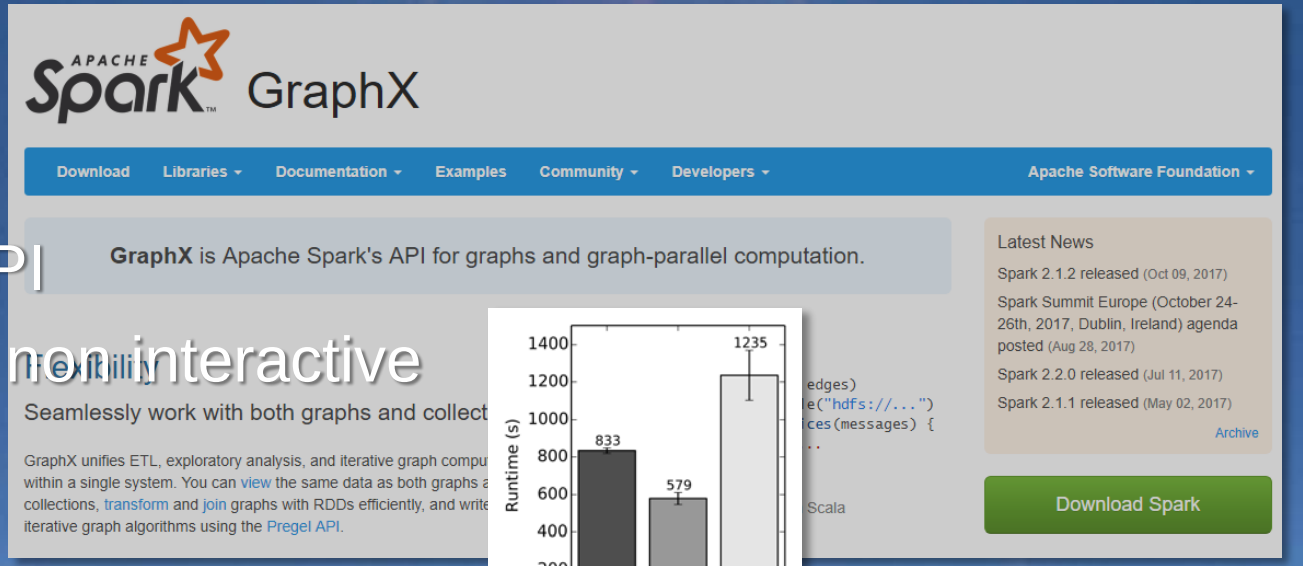
- Sometimes the problem is how to display the data
- Example: The “all black” case.
  - Use: jitter, alpha, density contour… hexbins!!



R Source (using ggplot2)

How to display

- For Spark
  - – GraphX
    - Pregel API
    - Fast, but non interactive
    - Doc
      - – API for
        - » Java
        - » Scala
        - » Python
        - » R (paquetes SparkR (Apache), sparklyr (RStudio), o graphframes (extension de sparklyr de RStudio))



How to display

- For R
  - bigvis package (by RStudio, not yet in CRAN despite started in 2013)
  - For datasets with 10-100 million observations (rows)
    - Create bins
    - Condense (manages data in bins)
    - Plot condense objects. Ubercool autoplot().
    - Smooth plot (to focus on main trends and hide outliers)
  - More info (pre-print paper)

How to display

- For R
  - bigvis package (by RStudio, not yet in CRAN despite started in 2013)

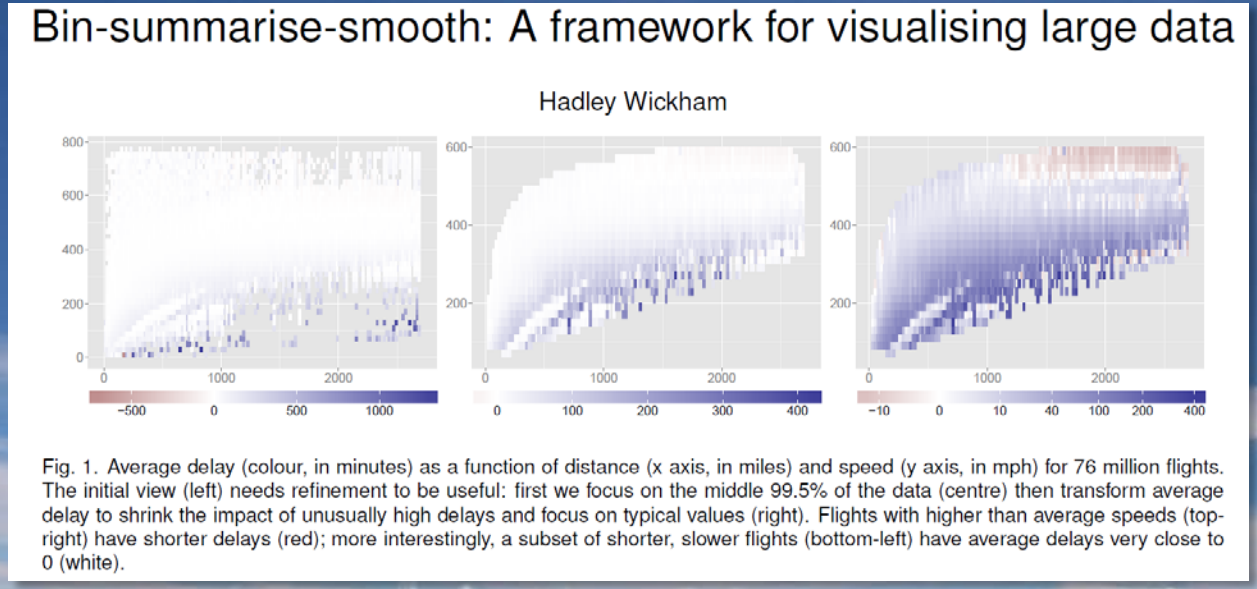
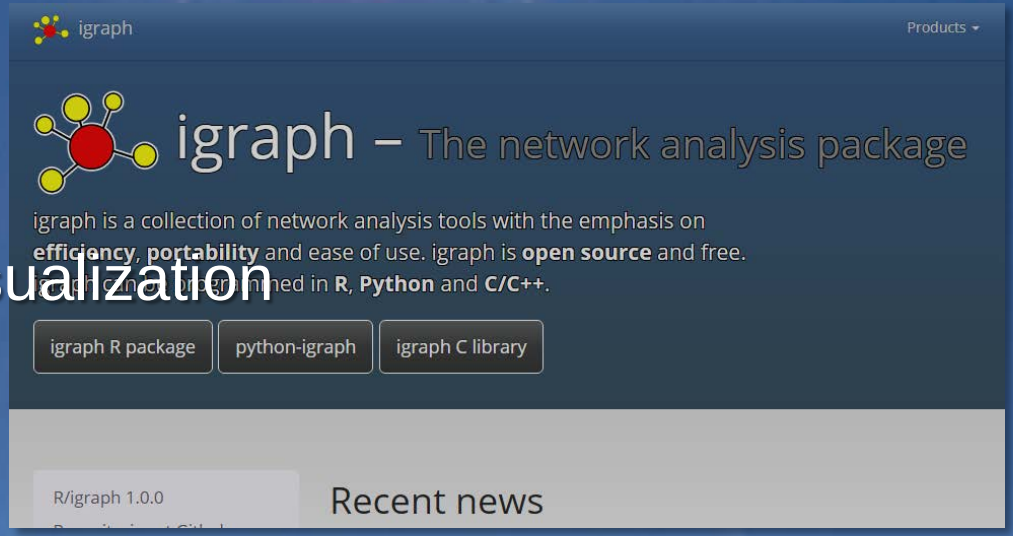


Fig. 1. Average delay (colour, in minutes) as a function of distance (x axis, in miles) and speed (y axis, in mph) for 76 million flights. The initial view (left) needs refinement to be useful: first we focus on the middle 99.5% of the data (centre) then transform average delay to shrink the impact of unusually high delays and focus on typical values (right). Flights with higher than average speeds (top-right) have shorter delays (red); more interestingly, a subset of shorter, slower flights (bottom-left) have average delays very close to 0 (white).

How to display networks

- For R
  - iGraph
    - Non interactive visualization
    - API for
      - C
      - Python
      - R
  - Have a look at the R Journal
    - E.g.: packages sna and networks (Butts 2014)



How to display networks

- For R
  - ggplot2 style for networks
    - See paper (R Journal 2017)
    - Functions
      - ggnet2 (in package GGally)
        - » Returns a ggplot2 object
          - » Different layers for labels, shapes, etc..
        - » Arguments
          - » an igraph/network
          - » a layout for the graph
          - » ggplot2 args

Functionality	ggnet2 (GGally)	geom_net (geomnet)	geom_nodes, geom_edges, etc (ggnetwork)
Data	object of class "network" or object easily converted to that class (i.e. incidence or adjacency matrices, edge list) or object of class "igraph"	a fortified "network", "igraph", "edgedf", or "adjmat" object OR one edge data frame and one node data frame to be merged internally	same as ggnet2
Naming conventions	node._, edge._, label._, edge.label._ for alpha, color, etc.	arguments identical to **ggplot2** with exception of ecolor, ealpha	same as **ggplot2**
Layout package & default	**sna**, Fruchterman-Reingold	**sna**, Kamada-Kawai	**sna**, Fruchterman-Reingold
Aesthetic mappings to variables	all alpha, color, shape, size for nodes, edges, labels	colour, size, shape, x, y, linetype, linewidth, label, group, fontsize	same as **ggplot2**
Arrows	directed = TRUE, arrow.size, gap	arrowsize, gap, arrow = arrow() like **ggplot2**	specify arrows in geom_edge like in code-geom_segment, arrow.gap
Theme or palette changes	done in the function with arguments like _.legend, _.palette, etc. and adding **ggplot2** elements	adding **ggplot2** elements	adding **ggplot2** elements
Creating small multiples	created separately, use grid.arrange from **gridExtra**	add group argument to fortify() and use facet_*() from **ggplot2**	use by argument in ggnetwork() and facet_*() from **ggplot2**
Edge labelling?	Yes	No	Yes
Draw self-loops?	No	Yes	No

Table 1: Comparing the three different package side-by-side.

How to display networks

- For R
  - ggplot2 style for networks
    - See paper (R Journal 2017)
    - Packages
      - ~~geomnet~~
        - ~~» A single layer~~
      - ggnetwork

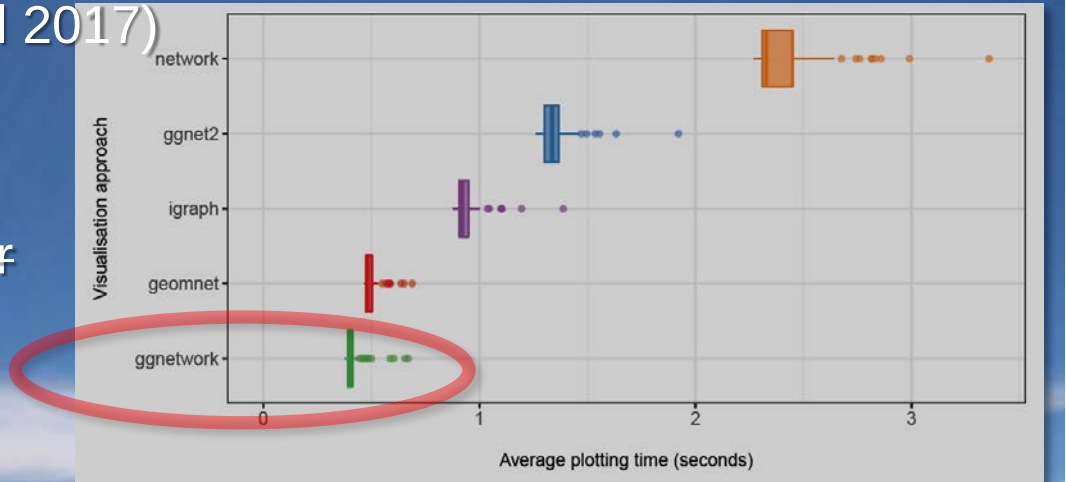
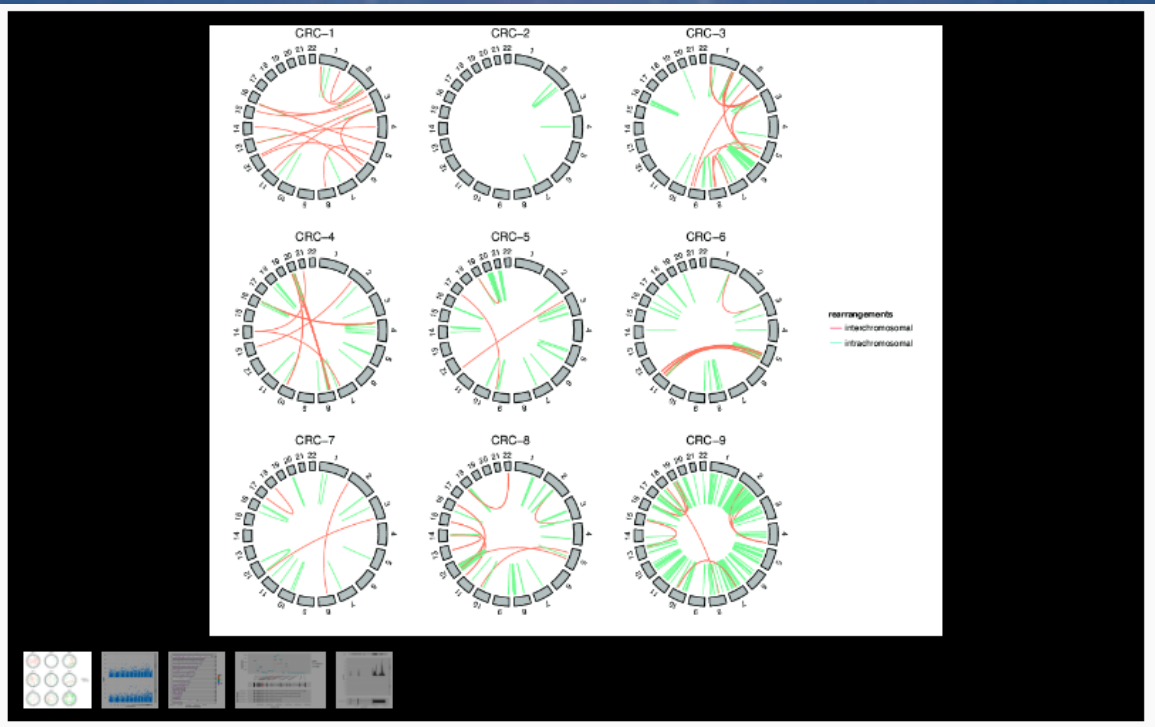


Figure 10: Comparison of the times needed for calculating and rendering the previously discussed protein interaction network in the three **ggplot2** approaches and the standard plotting routines of the **network** and **igraph** packages based on 100 evaluations each.

More ggplot2 style

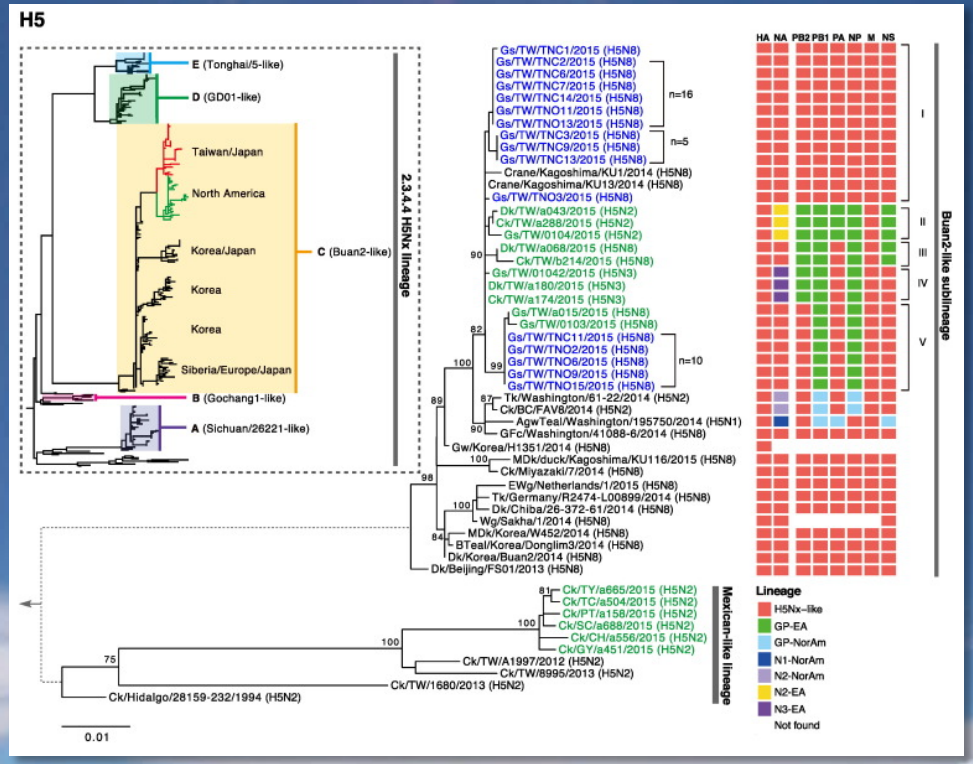
- ggbio



ggbio: An R implementation for extending the Grammar of Graphics for Genomic Data

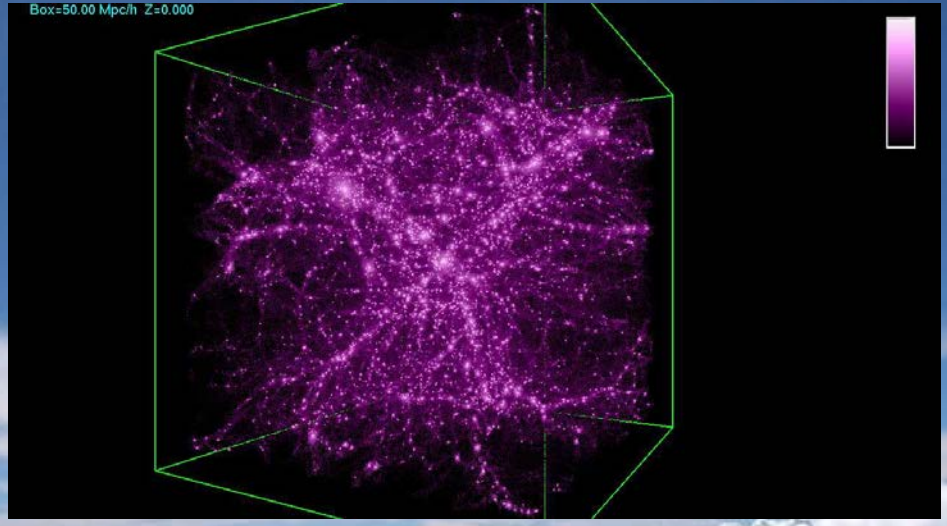
More ggplot2 style

- ggtree



Hardware limits

- Use GPUs
- Specific software
  - PMViewer rendering framework
    - Open source
    - 10 year old (aka “stable”)
    - Only for N-body simulations
    - See the gallery



What can I do with my PC?

- Aggregate points up to your screen resolution
- Use a modest rendering software
  - Which one? Wait for a while and see ☺

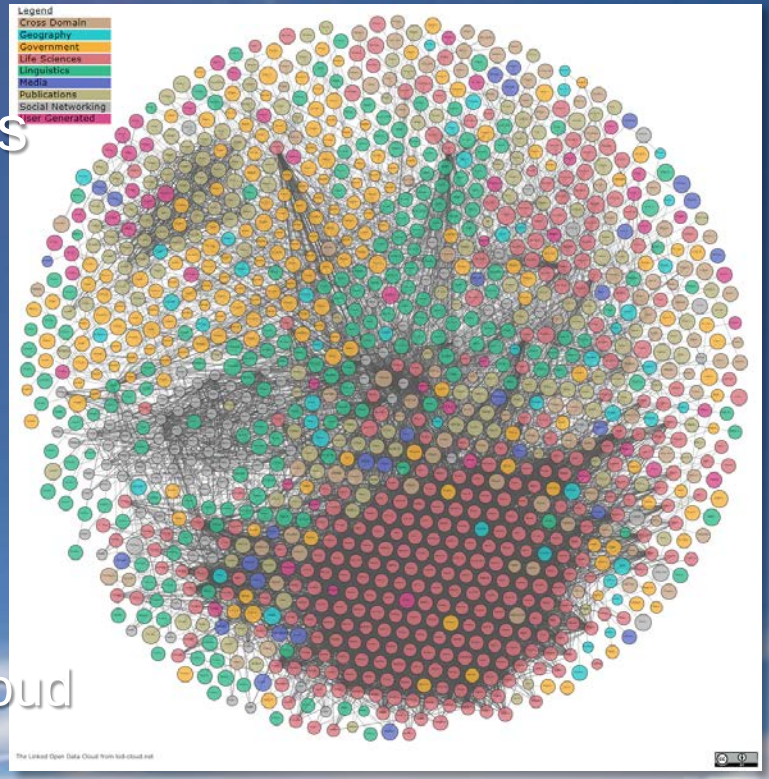
How to explore big datasets

BIG DATA USE CASE: LOD CLOUD

In the LOD galaxy…

LOD galaxy

- Numbers
  - Number of datasets: thousands
  - Number of triples: billions
    - DBpedia: 38M instances
      - 2.7K dbo properties (much more dbp)
    - Wikidata: 100M instances
      - 10K props
- How do we explore it?
  - The LOD (Linked Open Data) Cloud
    - 1349 datasets as of Dec 2024
      - » 16,000+ links



Big VISIBLE & INTERACTIVE graphs with

GEPHI

Gephi

- Big Graphs (up to ~1M nodes)
- 100% Java app (Windows, Mac, Linux)
- Open, free
- Java API
- Multicore algorithms
- Extensible (plugins). E.g.:
    - Graph import from SPARQL queries results

Gephi

Version 0.10.x (Apr 2022-today)
untested on Windows 10 & iOS
Doesn't support (yet)
SemanticWebImport ☹

- Installation
  - It is picky
    - Install Gephi version 0.10.1

Gephi

Version 0.9.2 (Sep 2017- Apr 2022) goes fine on Windows 10 & iOS

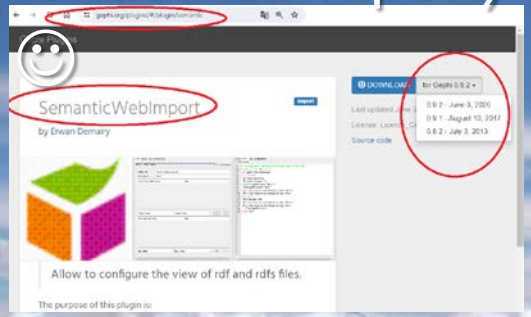
Version 0.9.1 (Feb 2016) has platform issues

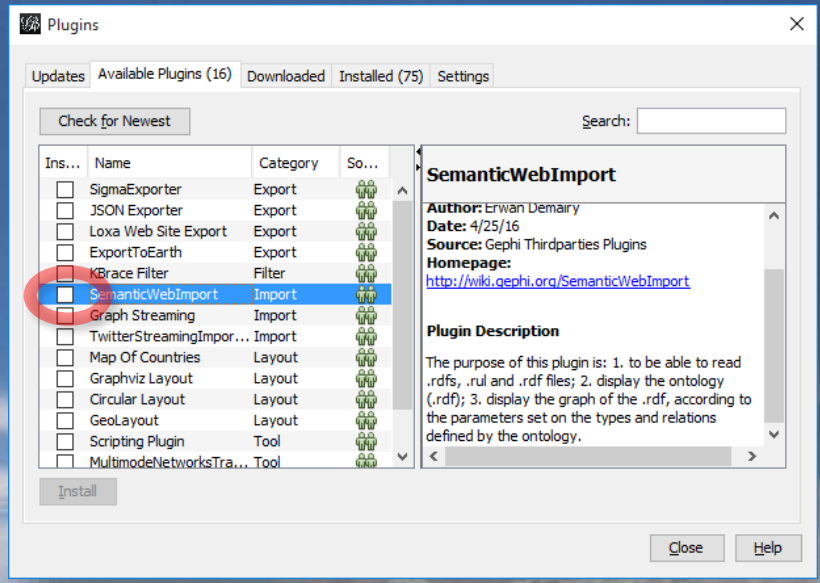
- Installation
  - It is picky
    - Install Gephi version 0.82 to support the WebImport plugin (latest version, 0.9, does NOT support it yet on all platforms)

      - Configure Gephi to use JDK (no JRE) SE 7 (Java 8 not supported yet). *Oracle ended support for Java 7 (You must be registered to download from here)*
        - » {gephiDir}/etc/gephi.conf is within a “read only” folder.
          - » Modify the file, save it to Desktop, move it to folder

Gephi

- 0.9.2 version Installation on Windows
  - Can run without **un**installing previous versions
  - The WebImport plugin (now SemanticWebImport) is listed



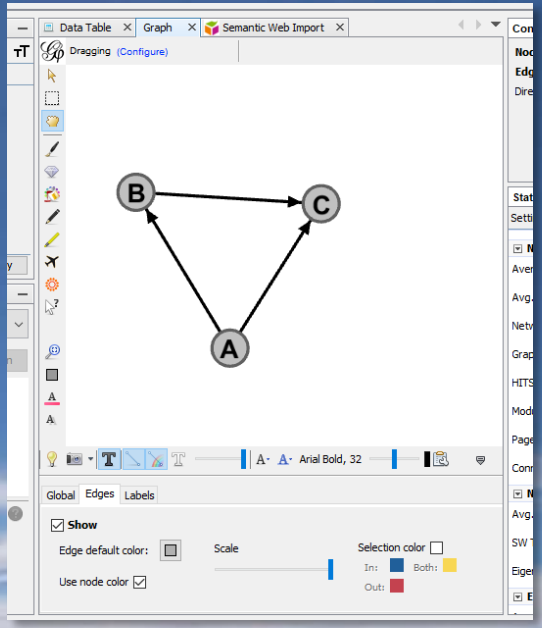


Gephi

- 0.92 version Installation on Mac (iOS)
  - It works. If you have issues, please, tell me ☺
- 0.91 version Installation on Mac (iOS)
  - You can install the plugin but IT DOES NOT WORK ☹

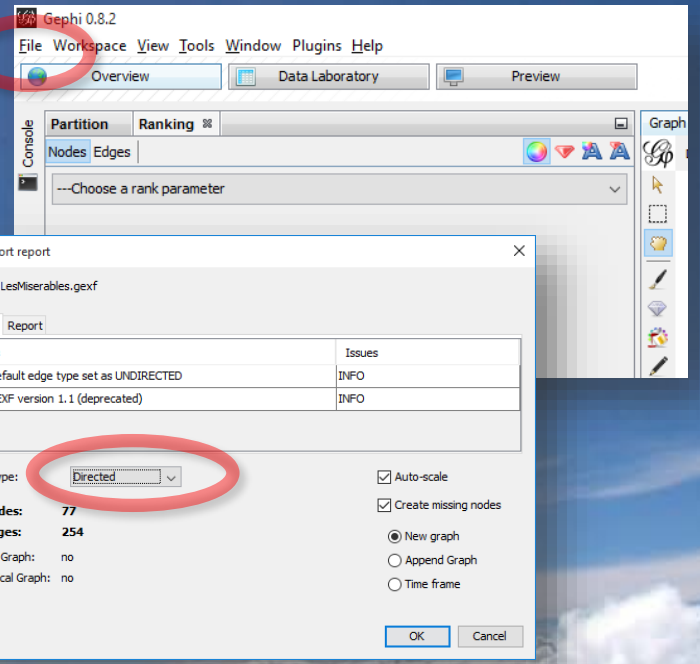
Hands on

- Type data manually in Gephi's table
  - Node A
  - Node B
  - Node C
  - Directed Edge A-B
  - Directed Edge A-C
  - Directed Edge B-C
- See the resulting graph
  - Zoom in/out using
    - "Two fingers" in touch pad
    - Mouse wheel



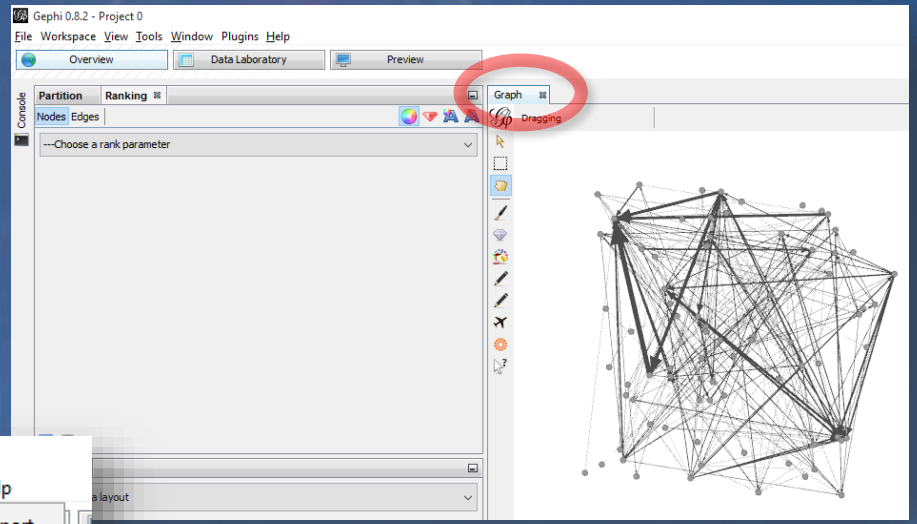
A not so quick tutorial

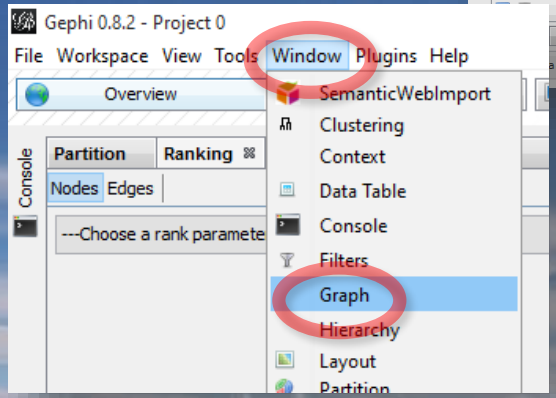
- Download a “gephi graph file”
  - LesMiserables.gexf (17 KB)
- Open Gephi
- Load LesMiserables.gexf
  - File → Open…
  - Use “undirected”



Quick tutorial

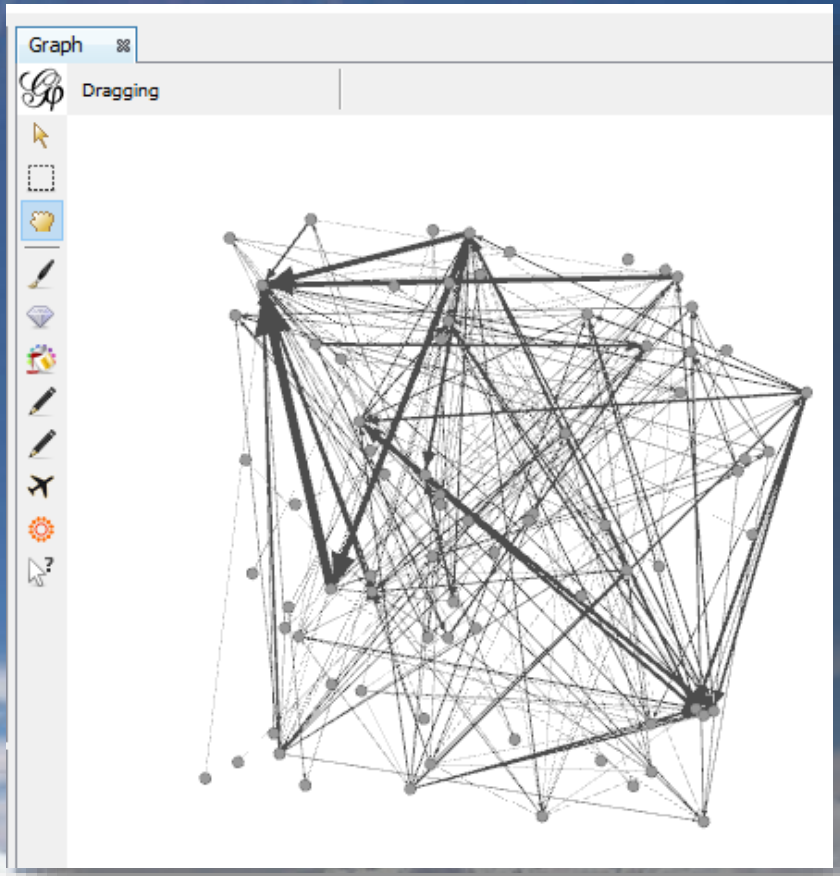
- If you do not see this
- Try this





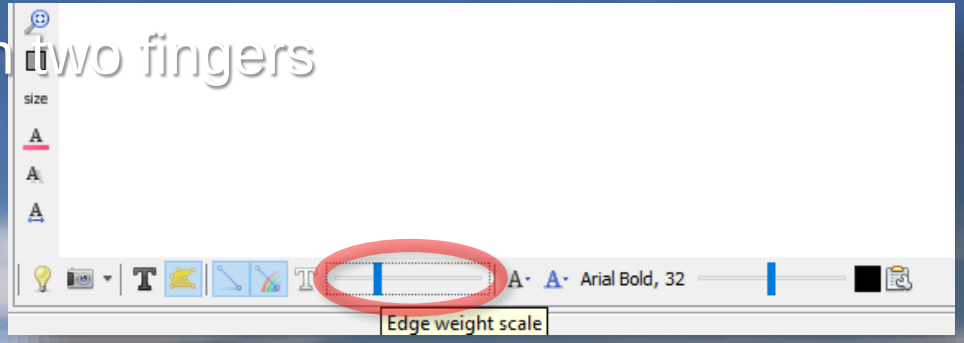
Quick tutorial

- Node positions are random. You will see a different graph
- Each node is a character of Victor Hugo's novel
- This graph is a "Co-appearance weighted network", that is,
  - Two vertices are adjacent if the corresponding characters encounter each other, in selected chapters of the novel



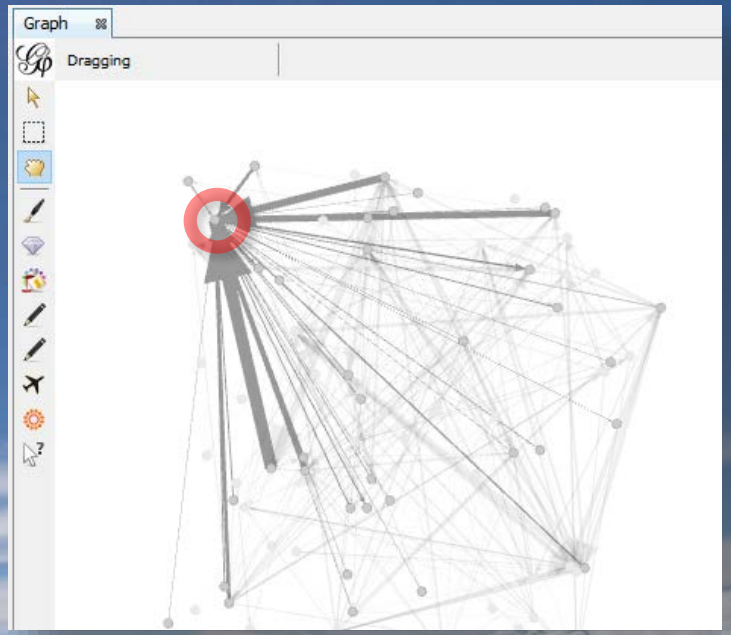
Quick tutorial

- Basics
  - Move the whole graph
    - Drag the mouse while **right** button is pressed
  - Zoom in/out
    - Use mouse wheel
    - or drag touchpad with two fingers
  - Width of edges (arcs)
    - Slider (no “text size” slider)



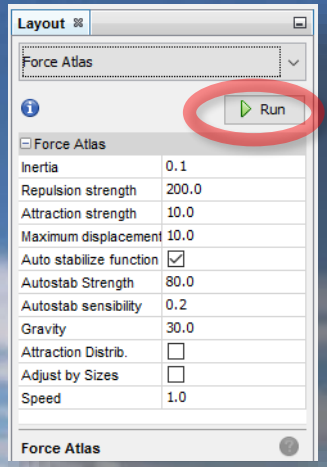
Quick tutorial

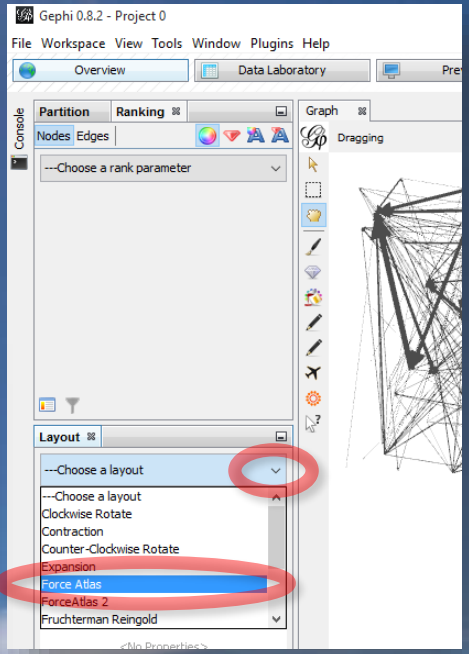
- Basics (cont.)
  - Hover (locate mouse pointer) over a node shows the links to/from that node
    - “Fading out” the rest of the graph



Quick tutorial

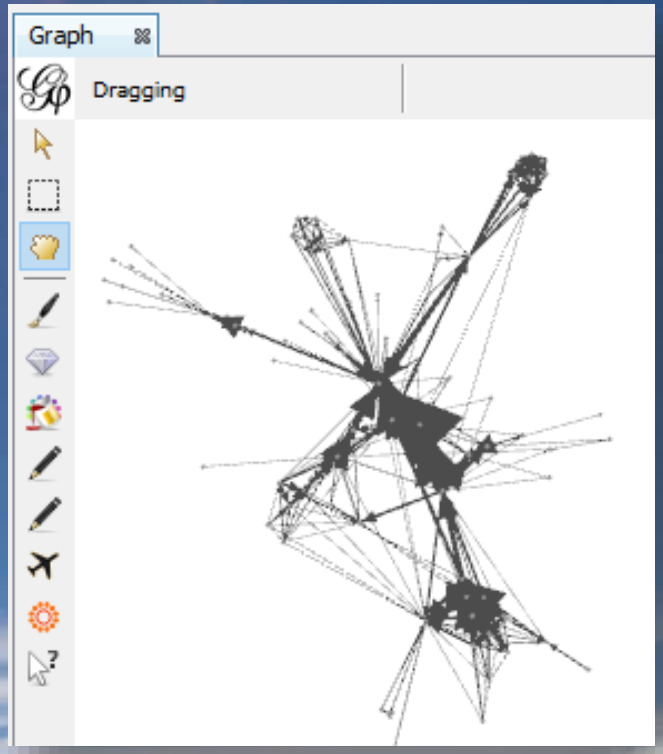
- Layout
  - Click on “Force Atlas”
    - This is a layout algorithm
      - Linked nodes “attract”, non-linked nodes are “repelled” (pushed apart)
  - Do not change default parameters
  - Press “Run” button





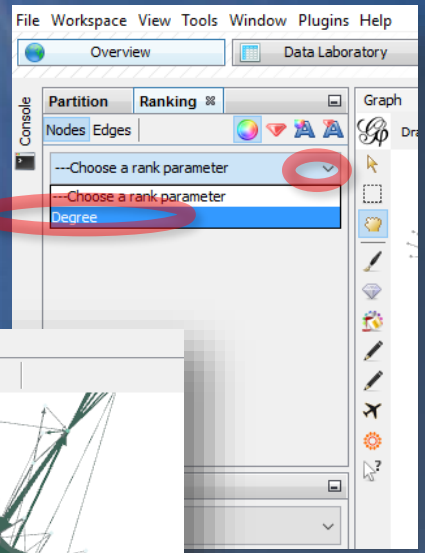
Quick tutorial

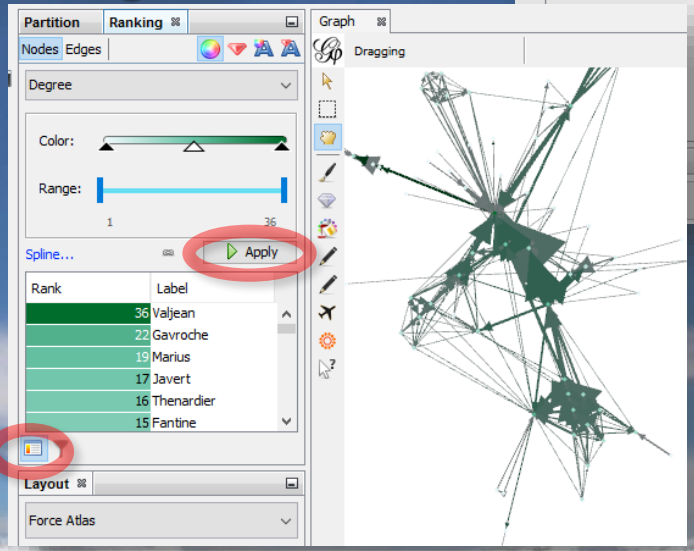
- Layout (cont.)
  - Press “Stop” button
  - Zoom in
  - Drag the graph
- You should get (something like) this



Quick tutorial

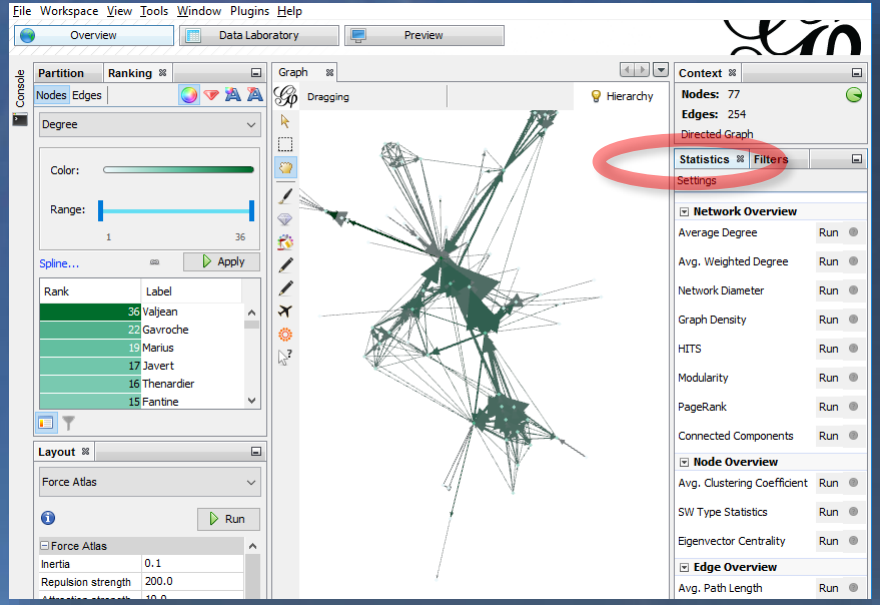
- Nodes' color
  - By default, the unique "ranking metric" is "Degree" (number of edges of each node)
    - You will get more metrics later on
  - Select "Degree"
  - Press "Apply"
  - Show rank values





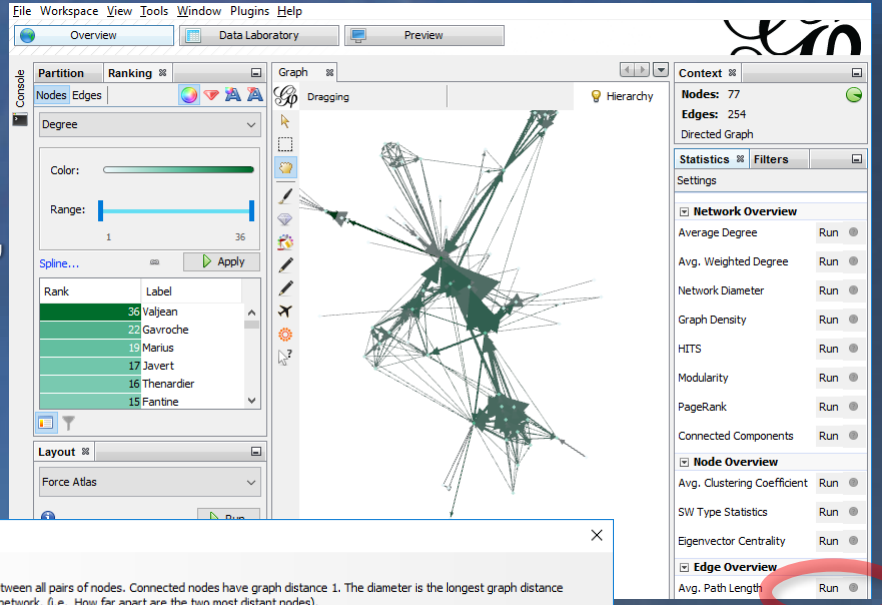
Quick tutorial

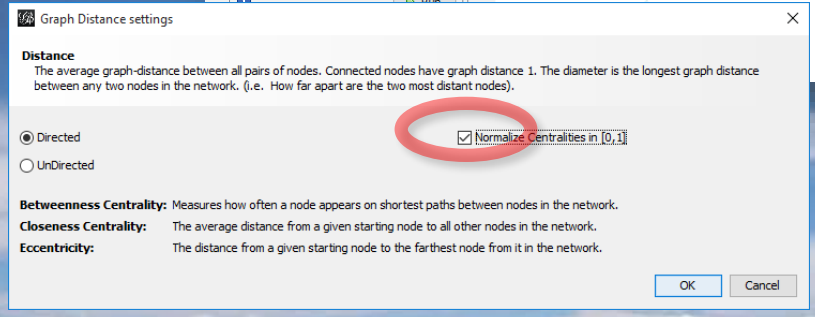
- Graph metrics
  - Many metrics (Statistics tab on the right side)
    - In 3 groups
      - Network overview
        - » Average Degree
        - » …
        - » Page Rank
      - Node overview
      - Edge overview



Quick tutorial

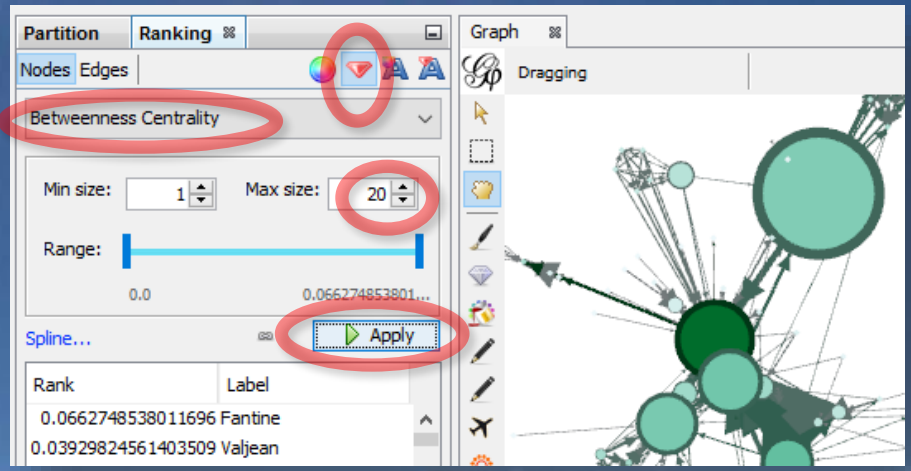
- Graph metrics (cont.)
  - In group “Edge overview” run the “Avg. Path length” method (press “Run” button)
  - 3 new metrics are computed:
    - Betweenness Centrality
    - Closeness Centrality
    - Eccentricity





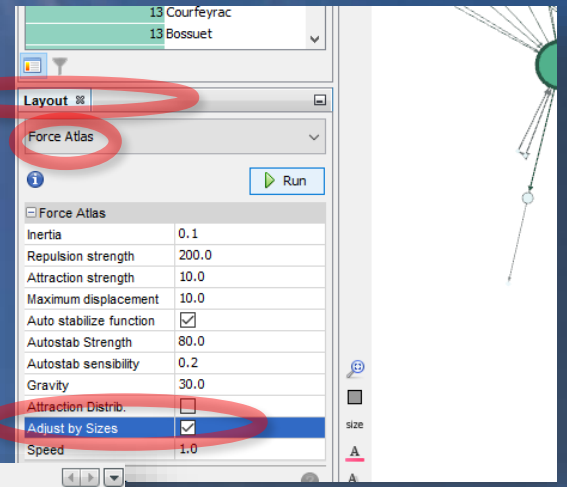
Quick tutorial

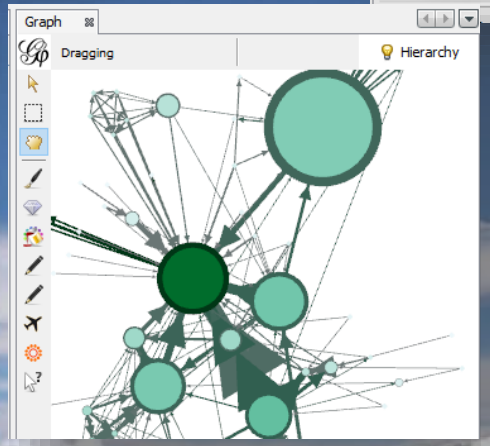
- Graph metrics (cont.)
  - Now you have more metrics in the "rank parameter"
  - Click node size icon
  - Select "Betweenness Centrality"
    - Higher values mean more influential (nodes); i.e. "bridges" between relevant nodes.
  - Apply (change "Max size") to apply this metric to nodes' size. Notice that node's color used "Degree" rank



Quick tutorial

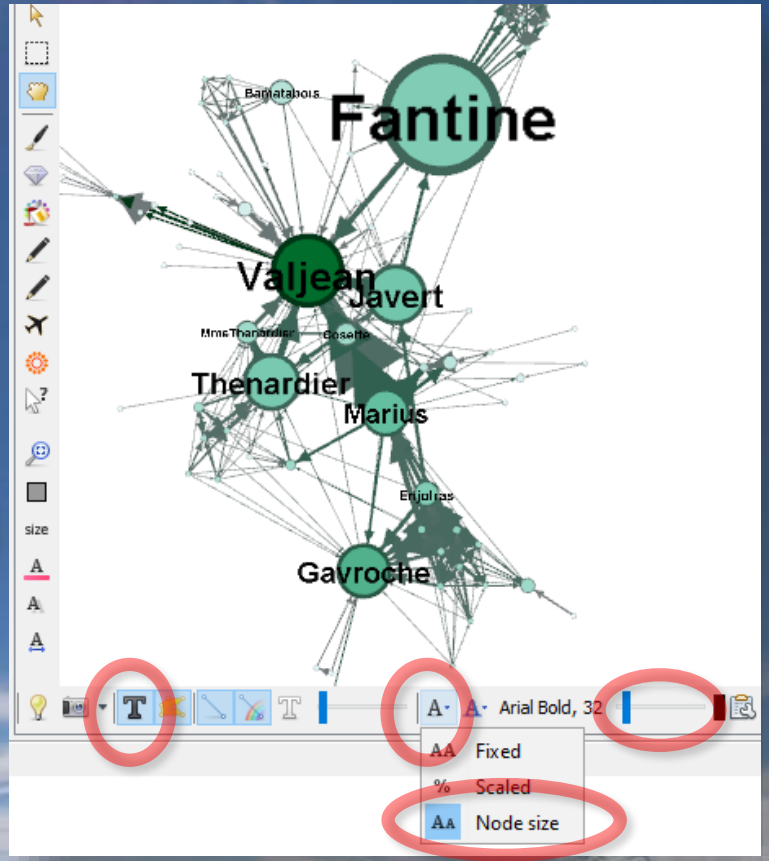
- Re-run layout algorithm avoiding nodes' overlaps
  - In "layout" tab
    - select "Force Atlas" algorithm
    - Click on "Adjust by Sizes"
    - Run for a while
  - You get (something like) this





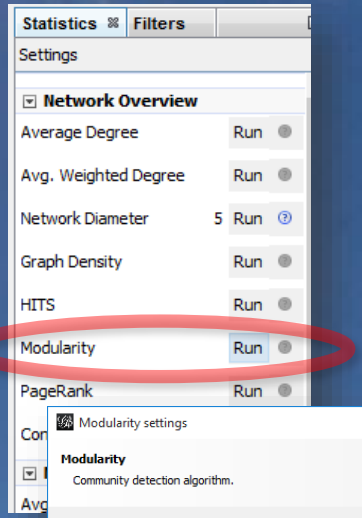
Quick tutorial

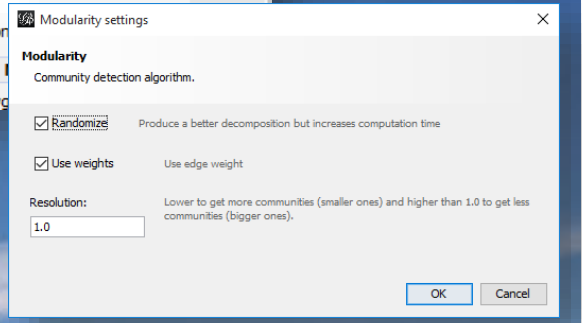
- Show Nodes' labels
  - Click on the T icon
  - Select "Node size" in the "size mode" selector
  - Move the "text size" slider
- You should get (something like) this

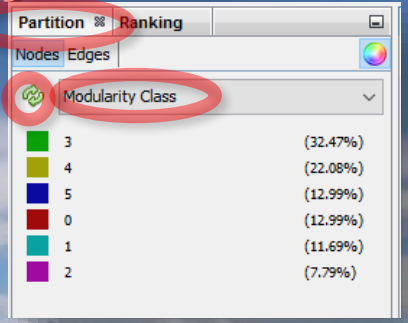


Quick tutorial

- Community detection
  - Run “modularity” in Statistics tab
    - With default parameters
  - Select “Partition” tab
  - Select “Modularity Class”
    - Refresh

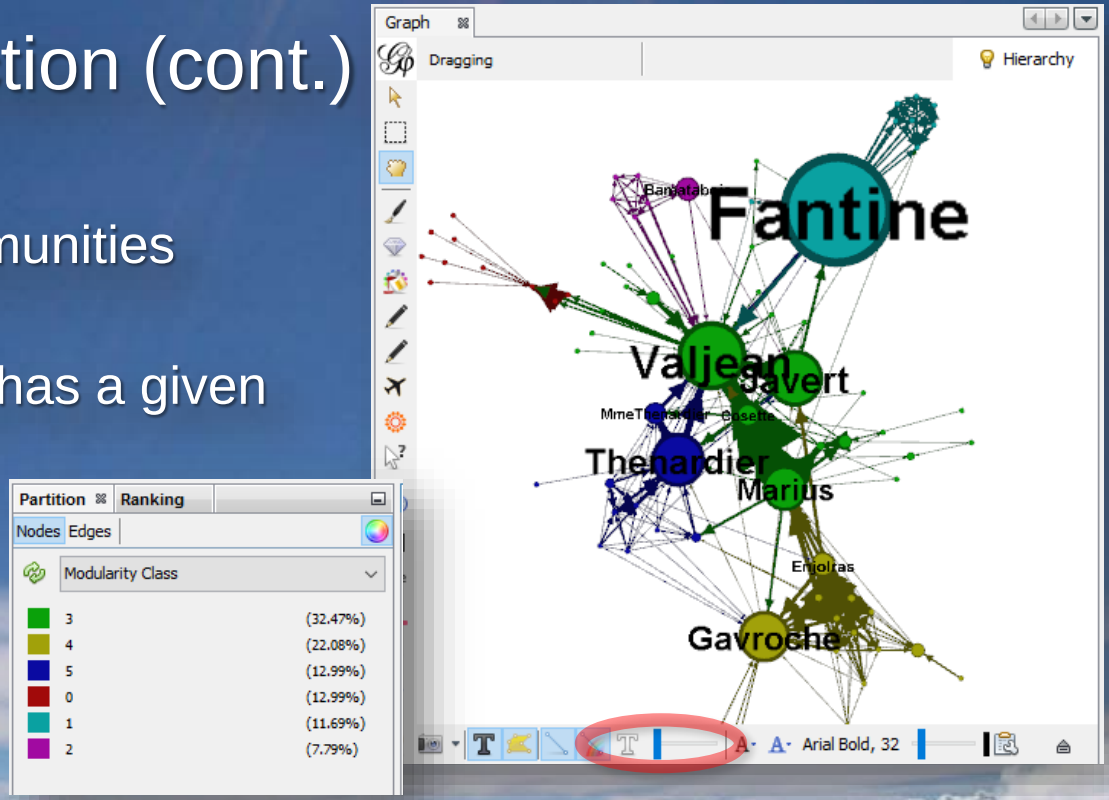






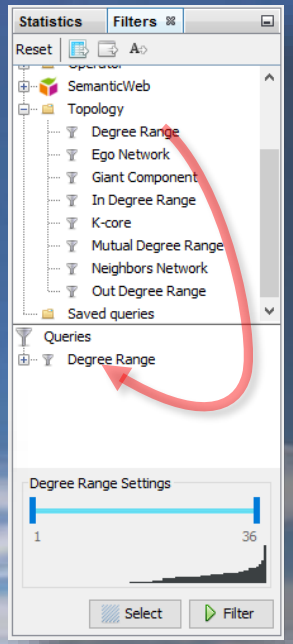
Quick tutorial

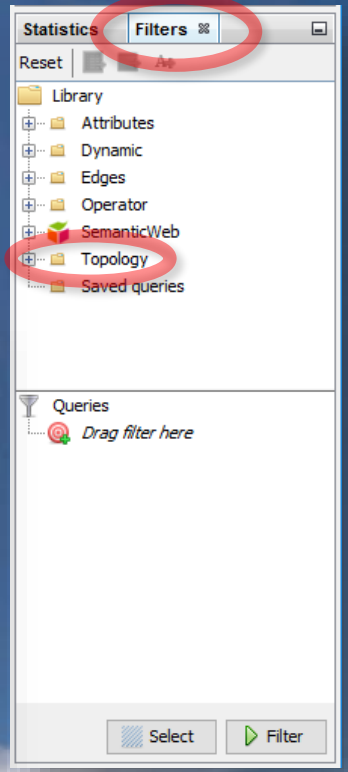
- Community detection (cont.)
  - You get this
    - There are 6 communities (clusters)
    - Each community has a given node/edge color
  - Reduce the edge weight scale
    - To get thinner edges



Quick tutorial

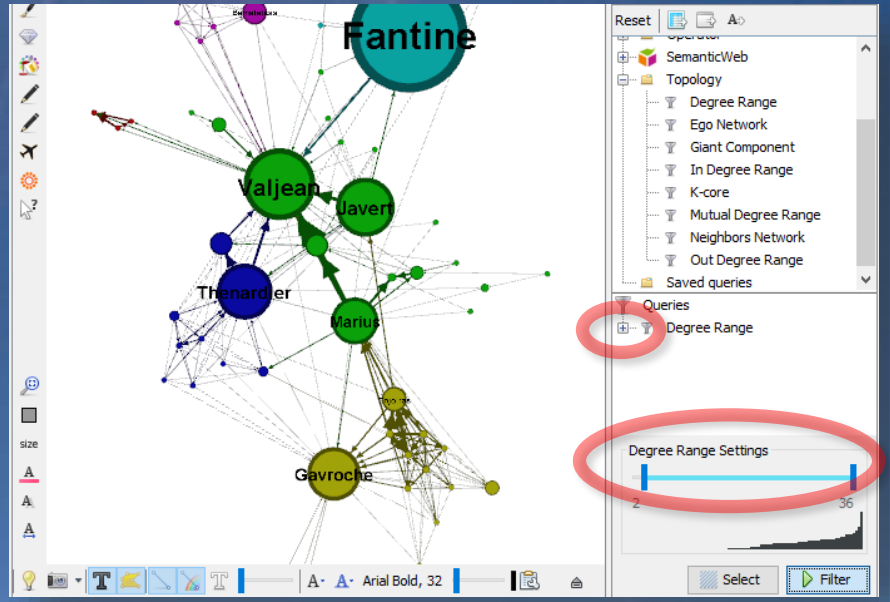
- Filters (cleaning up the graph)
  - Click on the “Filters” tab
  - Expand the “Topology” folder
  - Drag the “Degree Range” to the target icon 

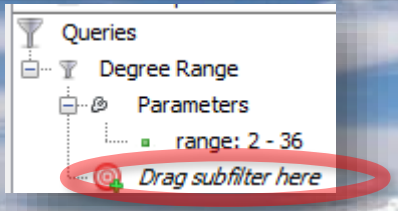




Quick tutorial

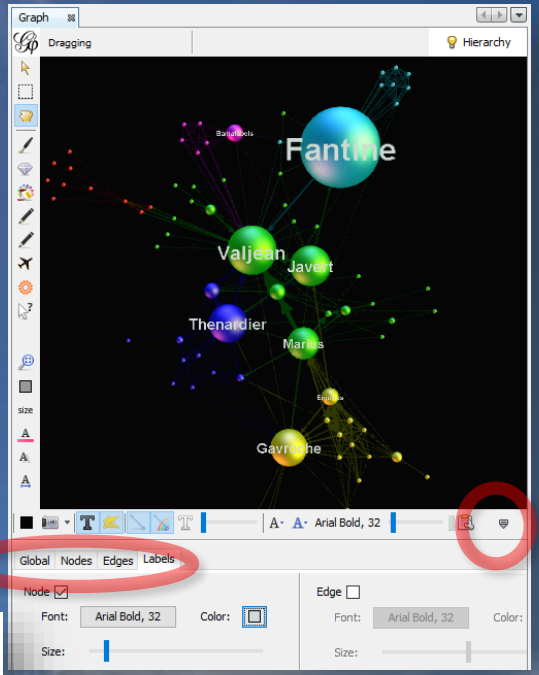
- Filters (cont.)
  - Move the slider to remove nodes with a degree lower than 2
  - Apply the filter
    - You get a cleaner graph
  - This query can be refined (expand the filter) with subfilters

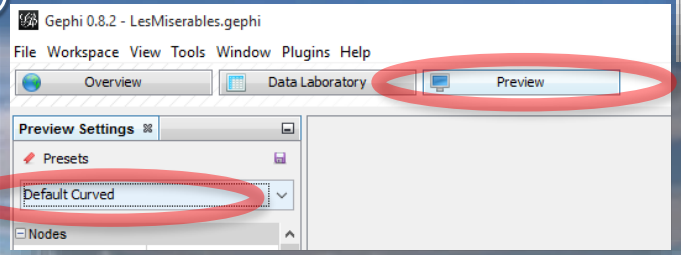




Quick tutorial

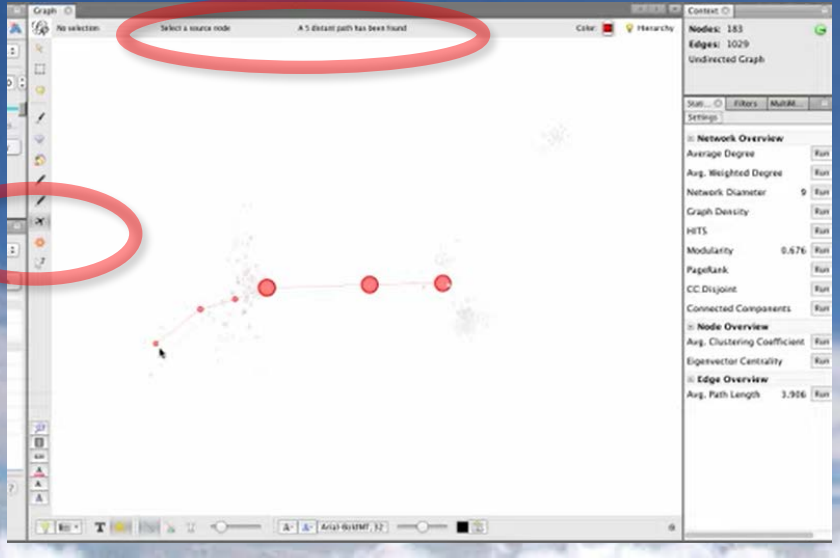
- Final output
  - Expand the “details” icon 
    - Assign values for bg color, etc.
  - Click the “Preview” window
    - I can not make it work on Windows ☹
      - But the file is OK





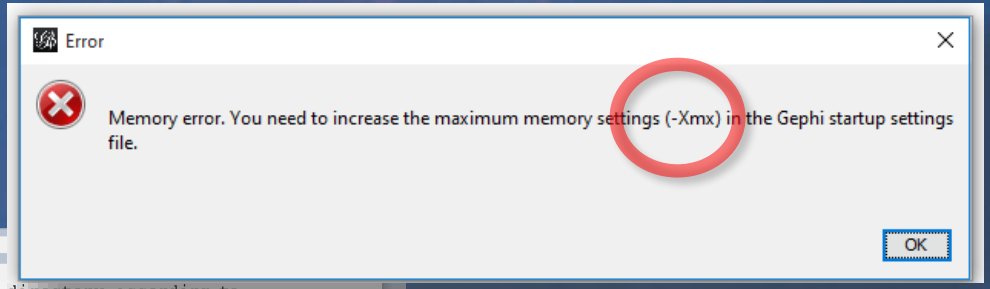
Extras

- Getting the shortest path in a graph
  - Use the “airplane icon”. Video here.



Gephi

- Out of memory
  - Edit {gephiDir}/etc/gephi.conf




```
# ${HOME} will be replaced by user home directory according to
platform
default_userdir="${HOME}/.${APPNAME}/0.8.2/dev"
default_mac_userdir="${HOME}/Library/Application Support/
${APPNAME}/0.8.2/dev"

# options used by the launcher by default, can be overridden
by explicit
# command line switches
default_options="--branding gephi -J-Xms64m -J-Xmx512m -J-
Xverify:none -J-Dsun.java2d.noddraw=true -J-
Dsun.awt.noerasebackground=true -J-
Dnetbeans.indexing.noFileRefresh=true -J-
Dplugin.manager.check.interval=EVERY_DAY -J-Duser.language=en"
# for development purposes you may wish to append: -J-
Dnetbeans.logger.console=true -J-ea

# default location of JDK/JRE, can be overridden by using --
jdkhome <dir> switch
jdkhome="C:\Program Files\Java\jdk1.7.0_79"

# clusters' paths separated by path.separator (semicolon on
Windows, colon on Unices)
#extra_clusters=
```

Big Linked Data graphs with

GEPHI LAYOUTS

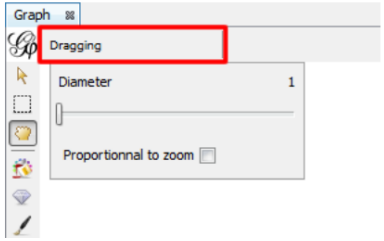
Gephi layouts

- See documentation at https://gephi.org/tutorials/gephi-tutorial-layouts.pdf
- Here I will show you a summary

Gephi layouts

Play against the algorithm!

Run the layout again and drag the nodes to stress it.



- Locate Dragging action, in the top left of the Visualization panel.
- Ajust the selection diameter in the panel or by using the shortcut “Ctrl + Mouse Wheel”.

Increase the “Autostab strength” in the Layout Properties to 100 000, then drag the nodes. The graph becomes less deformed.

- And now Stop the algorithm.

Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

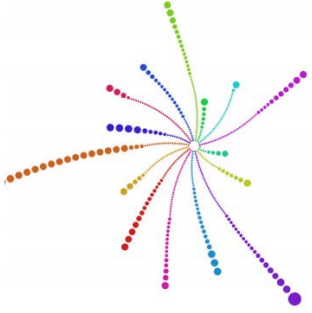
Various layouts exist

Gephi implements various layout algorithms. They set the shape of the graph.

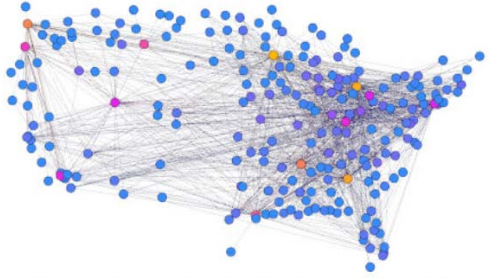
OpenOrd

ForceAtlas 2

Radial Axis



GeoLayout



Airlines sample dataset: http://gephi.org/datasets/airlines-sample.gexf

Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

So how to choose a layout?

In general, select one according to the feature of the topology you want to highlight:

emphasis DIVISIONS *OpenOrd*	emphasis COMPLEMENTARITIES *ForceAtlas, Yifan Hu, Frushterman-Reingold*
emphasis RANKING *Circular, Radial Axis*	emphasis GEOGRAPHIC REPARTITION *GeoLayout*

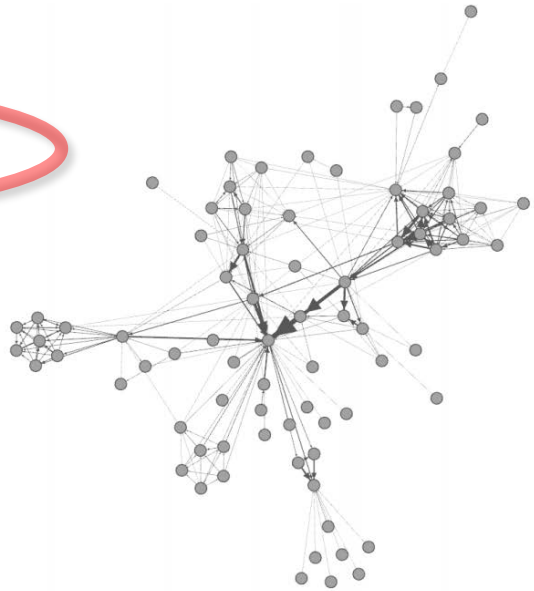
Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

ForceAtlas layout

Home-brew layout of Gephi, it is made to spatialize Small-World / Scale-free networks. It is focused on quality (meaning "being useful to explore real data") to allow a rigorous interpretation of the graph (e.g. in SNA) with the fewest biases possible, and a good readability even if it is slow.

Author:	Mathieu Jacomy
Date:	2007
Kind:	Force-directed
Complexity:	$O(N^2)$
Graph size:	1 to 10 000 nodes
Use edge weight:	Yes



Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

ForceAtlas layout

Home-brew layout of Gephi, it is made to spatialize Small-World / Scale-free networks. It is focused on quality (meaning "being useful to explore real data") to allow a rigorous interpretation of the graph (e.g. in SNA) with the fewest biases possible, and a good readability even if it is slow.

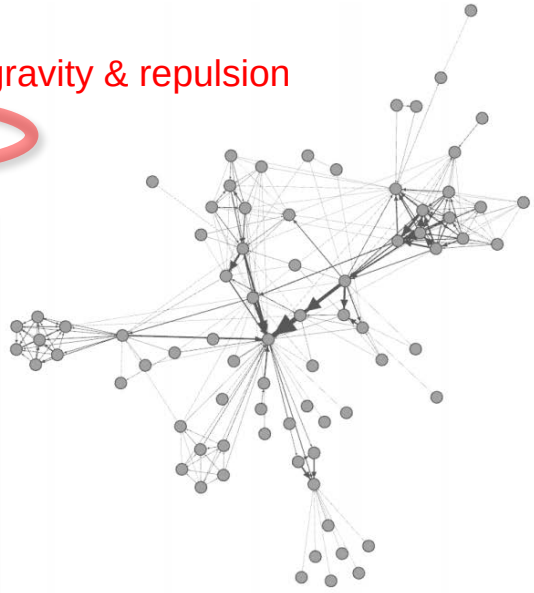
Author:	Mathieu Jacomy
Date:	2007
Kind:	Force-directed : gravity & repulsion
Complexity:	$O(N^2)$
Graph size:	1 to 10 000 nodes
Use edge weight:	Yes

Run ForceAtlas

Run the layout by applying the following settings step by step:

• Autostab strength = 2 000	Increase to move the nodes slowly.
• Repulsion strength = 1 000	How strongly does each node reject others.
• Attraction strength = 1	How strongly each pair of connected nodes attract each other.
• Gravity = 100	Attract all nodes to the center to avoid dispersion of disconnected components.
• Attraction Distrib. = checked	Push hubs (high number of output links) at the periphery and put authorities (high number of input links) more central.

And now Stop the algorithm.



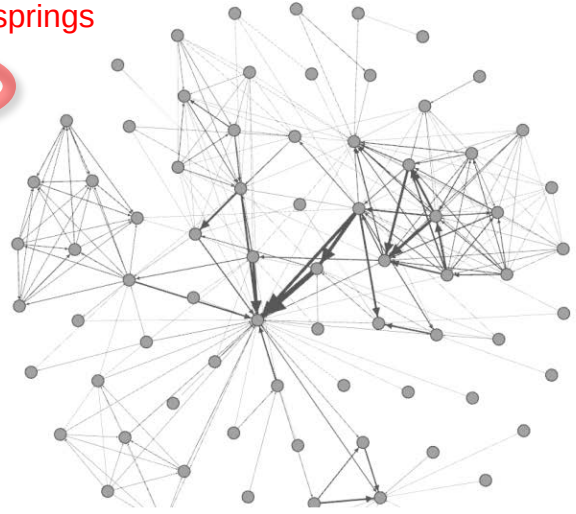
Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

Fruchterman-Reingold layout

It simulates the graph as a system of mass particles. The nodes are the mass particles and the edges are springs between the particles. The algorithms try to minimize the energy of this physical system. It has become a standard but remains very slow.

Author:	Thomas Fruchterman & Edward Reingold[1]
Date:	1991
Kind:	Force-directed: springs
Complexity:	$O(N^2)$
Graph size:	1 to 1 000 nodes
Use edge weight:	No



Run Fruchterman-Reingold

Run the layout by applying the following settings step by step:

• Area = 100 • Area = 100 000	Graph size area.
• Gravity = 1 000 • Gravity = 100	Attract all nodes to the center to avoid dispersion of disconnected components.

And now Stop the algorithm.

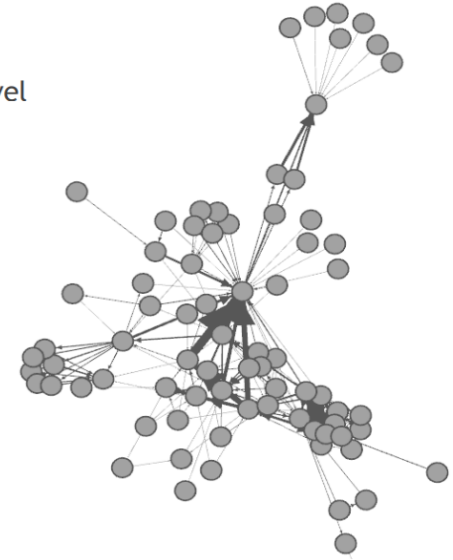
Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

Yifan Hu Multilevel layout

It is a very fast algorithm with a good quality on large graphs. It combines a force-directed model with a graph coarsening technique (multilevel algorithm) to reduce the complexity. The repulsive forces on one node from a cluster of distant nodes are approximated by a Barnes-Hut calculation, which treats them as one super-node. It stops automatically.

Author:	Yifan Hu[1]
Date:	2005
Kind:	Force-directed + multilevel
Complexity:	O(N*log(N))
Graph size:	100 to 100 000 nodes
Use edge weight:	No



Run Yifan Hu Multilevel

Launch the layout by applying the following settings step by step:

- Step ratio = 0.99 — Run — Ratio used to update the step size. Increase it for a better quality (vs speed).
- Optimal distance = 200 — Run — Natural length of the springs. Increase it to place nodes farther apart.
- Theta = 1.0 — Run — Approximation for Barnes-Hut calculation. Smaller values mean more accuracy.

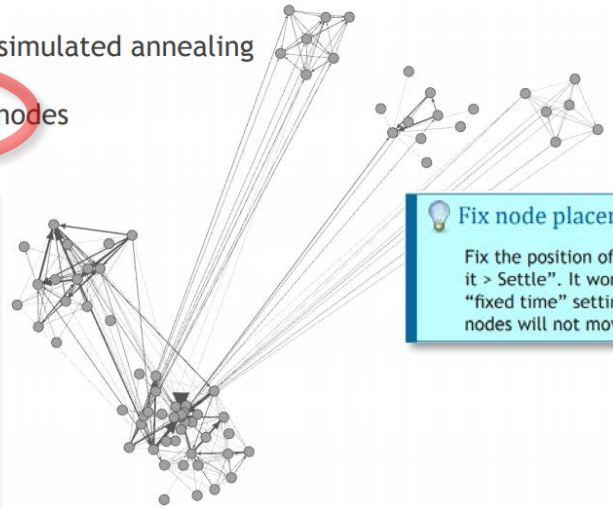
Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

OpenOrd layout

It expects undirected weighted graphs and aims to better distinguish **clusters**. It can be run in parallel to speed up computing, and stops automatically. The algorithm is originally based on Frutcherman-Reingold and works with a fixed number of iterations controlled via a simulated annealing type schedule (liquid, expansion, cool-down, crunch, and simmer). Long edges are cut to allow clusters to separate.

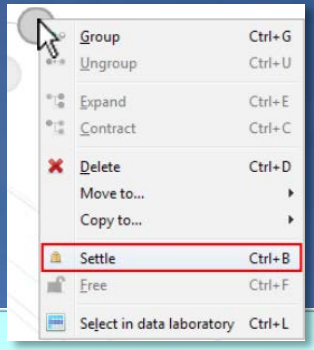
Author:	S. Martin, W. M. Brown, R. Klavans, and K. Boyack[1]
Date:	2010 (VxOrd)
Kind:	Force-directed + simulated annealing
Complexity:	O(N*log(N))
Graph size:	100 to 1 000 000 nodes
Use edge weight:	Yes



Run OpenOrd

Launch the layout by applying the following settings step by step:

- Edge cut = 0.95 — Run — From 0 (standard Frutcherman-Reingold) to 1. Percentage of the greatest distance between two nodes in the drawing. A higher cutting means a more clustered result.
- Num iterations = 100 — Run
- Num iterations = 850 — Run — Contract the clusters. Expand the clusters.
- Random seed = -6308261588084905834 — Run — Use this value to produce exactly the same shape as shown before.



Fix node placement

Fix the position of a node (or a group of selected nodes) by "Right-click on it > Settle". It works for all layouts except Yifan Hu. For OpenOrd, use the "fixed time" setting on the Layout panel to configure the time the fixed nodes will not move.

Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

Gephi layouts

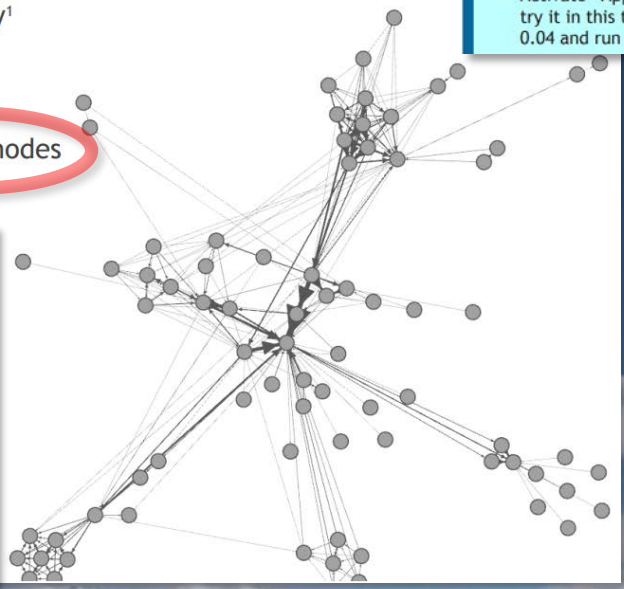
ForceAtlas 2 layout

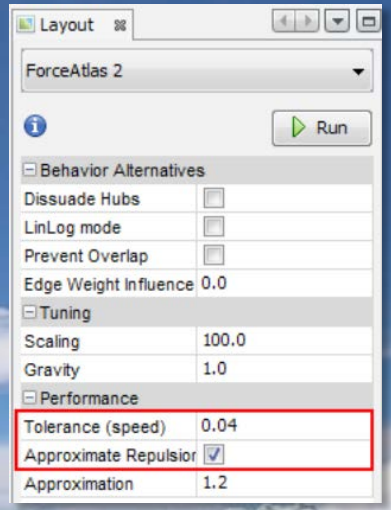
Improved version of the Force Atlas to handle large networks while keeping a very good quality. Nodes repulsion is approximated with a Barnes-Hut calculation, which therefore reduces the algorithm complexity. Replace the “attraction” and “repulsion” forces by a “scaling” parameter.

Author:	Mathieu Jacomy[1]
Date:	2011
Kind:	Force-directed
Complexity:	O(N*log(N))
Graph size:	1 to 1 000 000 nodes
Use edge weight:	Yes

Performance

Activate “Approximate Repulsion” on large graphs only, but let’s try it in this tutorial. Check it, set the “Tolerance” option to 0.04 and run the algorithm to see how nodes are swinging!





Run ForceAtlas 2

Run the layout by applying the following settings step by step:

• LinLog mode = checked • LinLog mode = unchecked	Linear attraction & logarithmic repulsion (lin-lin by default), makes clusters tighter.
• Scaling = 100	Increase to make the graph sparser.
• Edge weight influence = 0	From 0 (no influence) to 1 (normal). Set 0 to calculate forces without edge weight.

And now Stop the algorithm.

Source: https://gephi.org/tutorials/gephi-tutorial-layouts.pdf

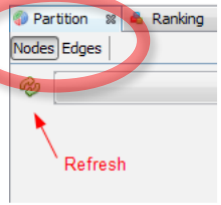
Gephi layouts

Detect communities

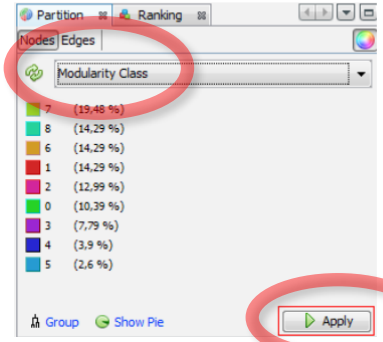
We now want to study the community structure in this network: does it divide naturally into groups of nodes with dense connections within groups and sparser connections between groups?

In the Statistics panel, click on Run near the "Modularity"[1] line.

The community detection algorithm created a "Modularity Class" value for each node. The partition module can use this new data to colorize communities.



- Locate the Partition module on the left panel.
- Click on the "Refresh" button to populate the partition list.



- Select "Modularity Class" in the partition list.

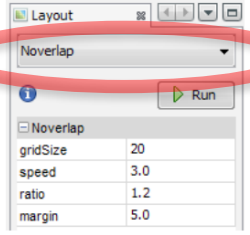
You can see that 9 communities were found, could be different for you. A random color has been set for each community identifier.

- Click on Apply to colorize nodes.

Gephi layouts

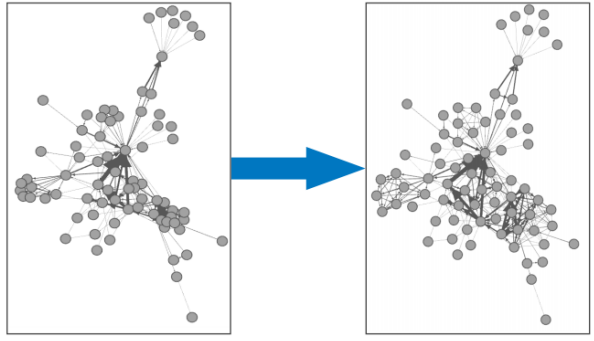
Noverlap layout

Use it after any layout to prevent node overlap while keeping the shape of the graph. It is optimized for big graphs.



- First, run the “YifanHu” layout.
- Select the “Noverlap” algorithm and run it until it stops.
- Reduce the “speed” setting to 0.1 to increase quality.
- Increase the “ratio” at 2 and “margin” at 10 for more spacing around nodes.

You can see nodes are not overlapping anymore.

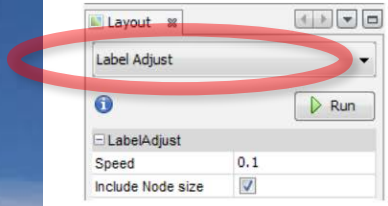


Gephi layouts

Label Adjust layout

It works on text size to repulse nodes and therefore makes every label readable. It only runs on the visible nodes in the Visualization panel.

- Locate the Visualization settings.
- Click on T to activate text display.
- Increase the text size to the maximum.



- Go to the Layout panel.
- Select the "Label Adjust" algorithm and run it until it stops.

Big Linked Data graphs with

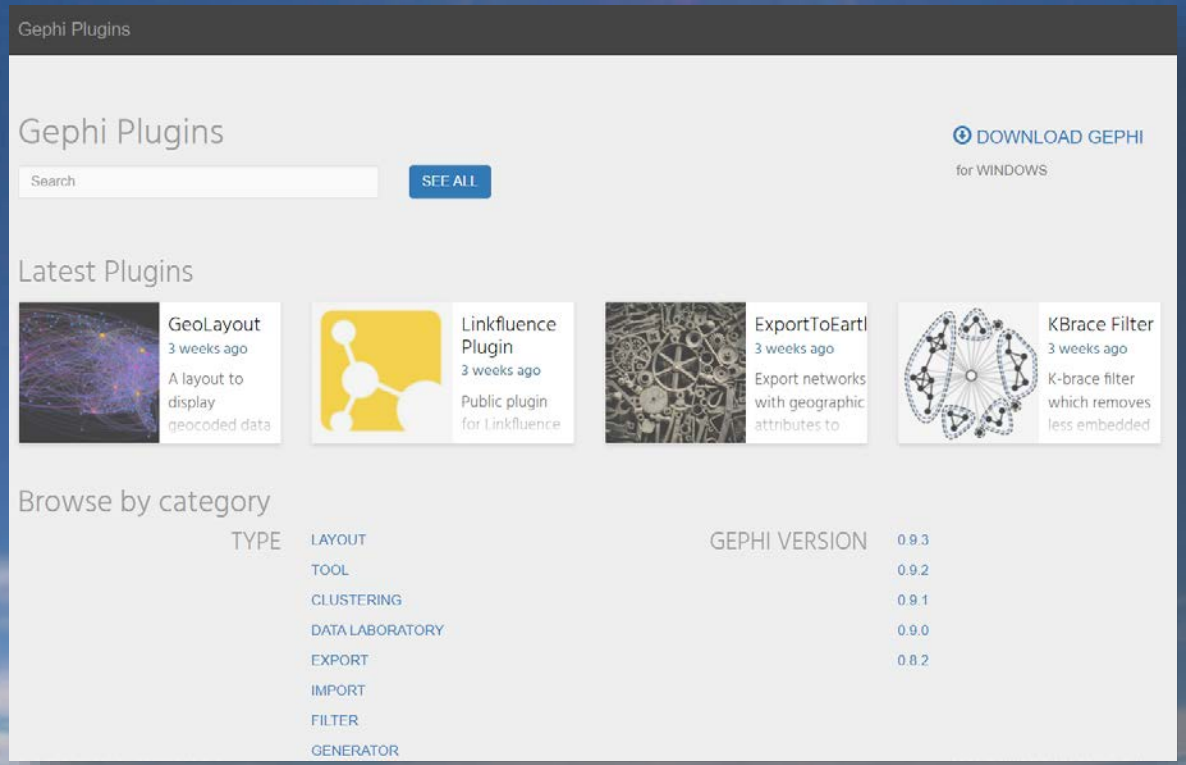
GEPHI + DBPEDIA (OR ANY OTHER SPARQL EP)

Gephi

- Plugins mechanish
  - SemanticWebImport (for Gephi 0.9.2) not yet for versión 0.9.5 or 0.10
  - ~~Virtuoso Importer (for Gephi 0.8.2)~~
  - Gephi Graph Streaming
  - Neo4J Graph Database Support

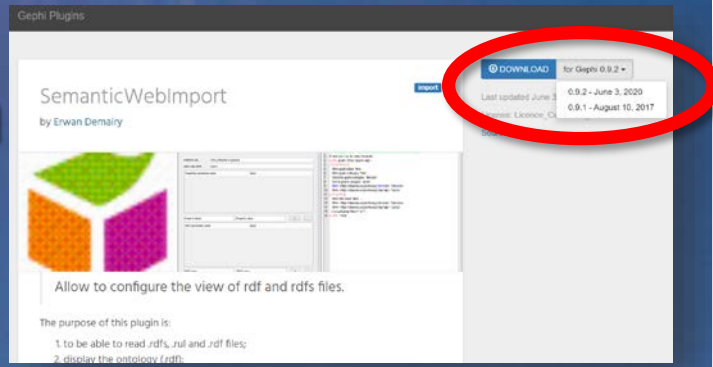
Gephi

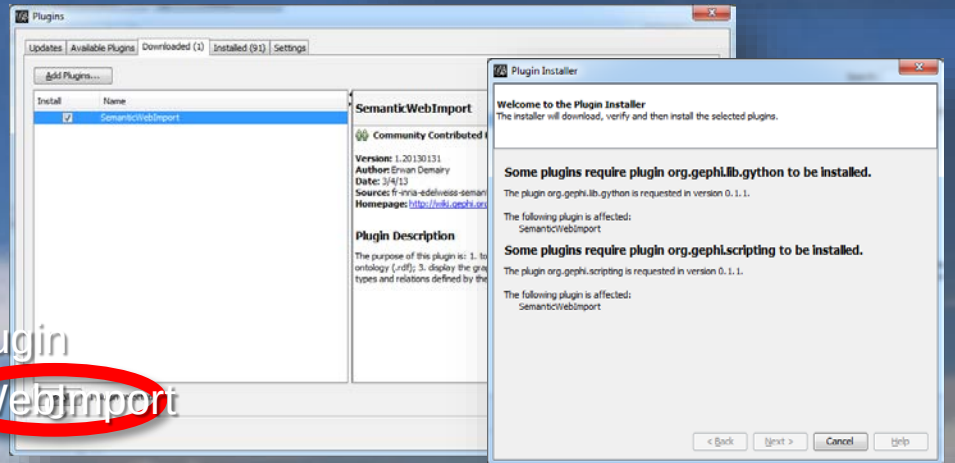
- Plugins



Gephi

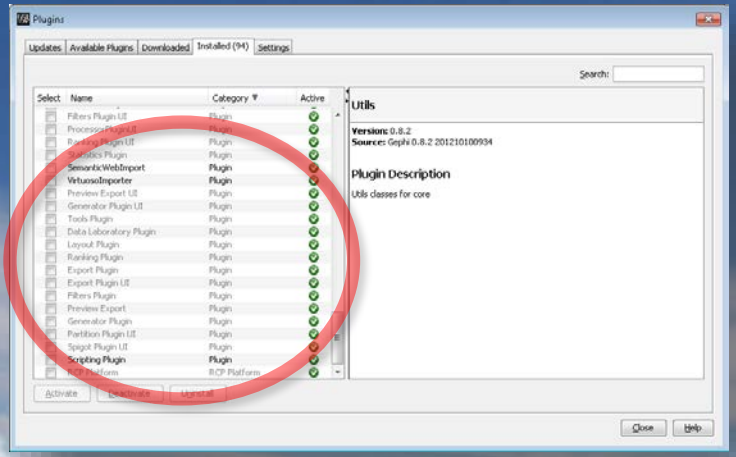
- SemanticWebImport plugin
  - In the Plugins section
  - Download it as a nbm file
  - Install it in Gephi
    - Tools→plugins→ Downloaded → Add Plugins
      - ??Firstly: Scripting plugin
      - Secondly: SemanticWebImport

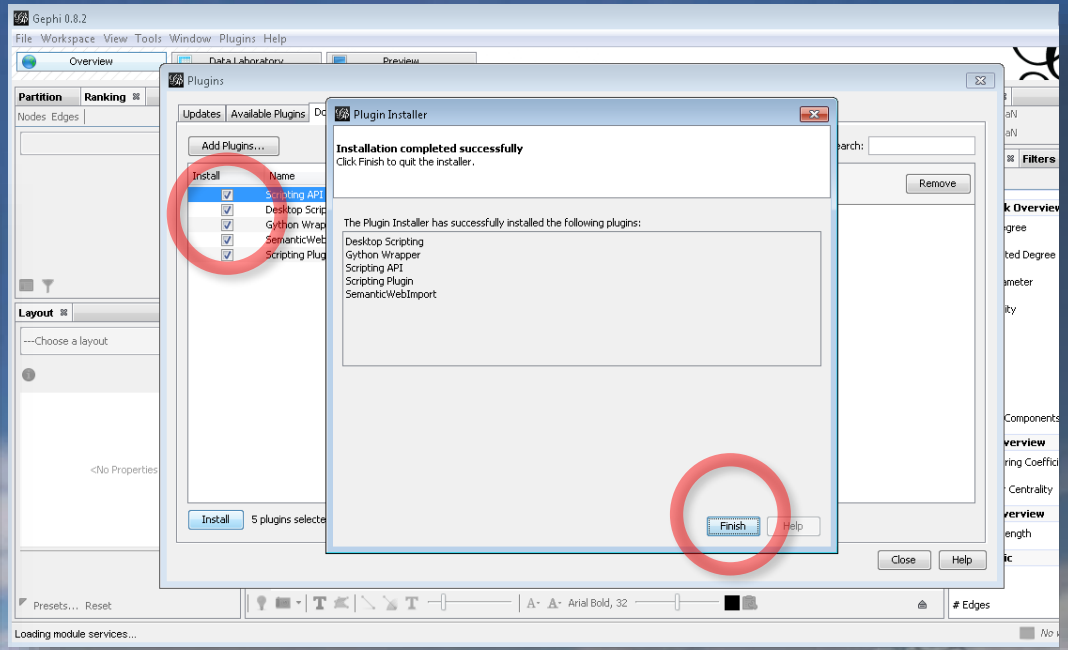




Gephi

- Loading the Scripting Plugin
  - zip file →
    4 nbm files

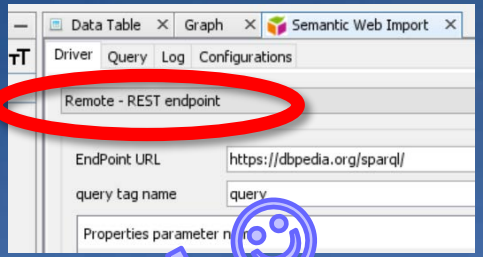




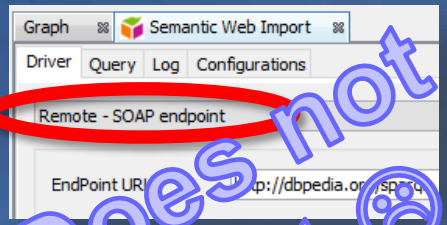
Gephi 0.9.2

- Importing data from DBpedia
  - 1) Point to (Driver tab) http**S**://dbpedia.org/sparql**/**

Pay attention to the ending /

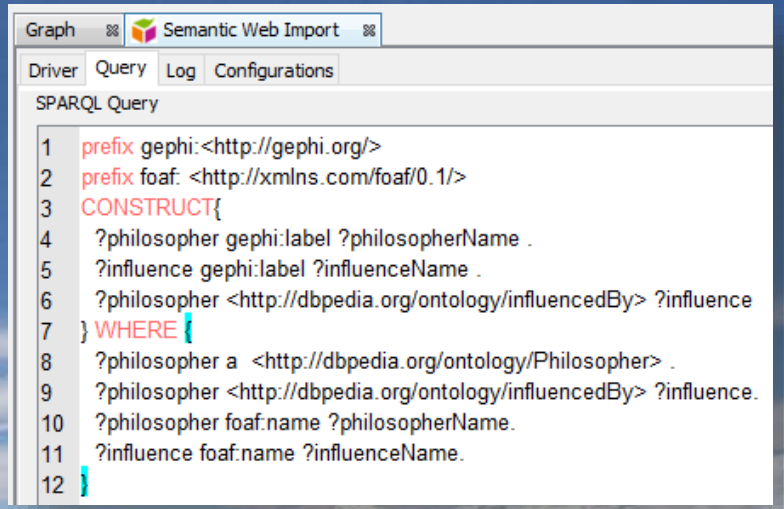


Yes! ☺



Does not exist ☹

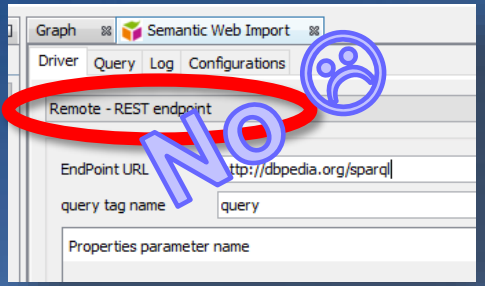
  - 2) Make a SPARQL **CONSTRUCT** query to relate gephi vars with SPARQL vars

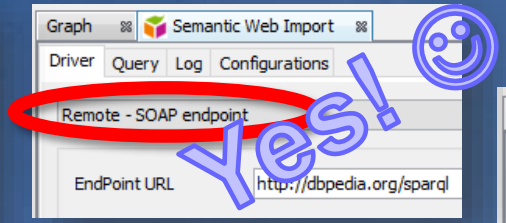



```
prefix gephi:<http://gephi.org/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT{
  ?philosopher gephi:label ?philosopherName .
  ?influence gephi:label ?influenceName .
  ?philosopher <http://dbpedia.org/ontology/influencedBy> ?influence
} WHERE {
  ?philosopher a  <http://dbpedia.org/ontology/Philosopher> .
  ?philosopher <http://dbpedia.org/ontology/influencedBy> ?influence.
  ?philosopher foaf:name ?philosopherName.
  ?influence foaf:name ?influenceName.
}
```

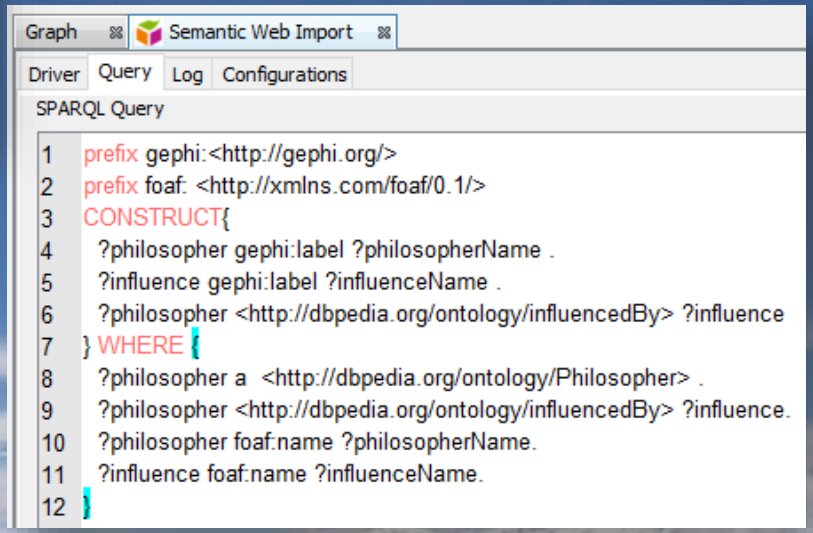
Gephi 0.8.2

- Importing data from DBpedia
  - 0) Create a new (empty) workspace/project. IMPORTANT!!



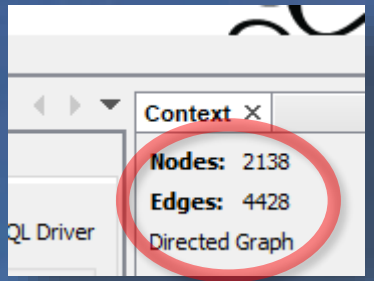


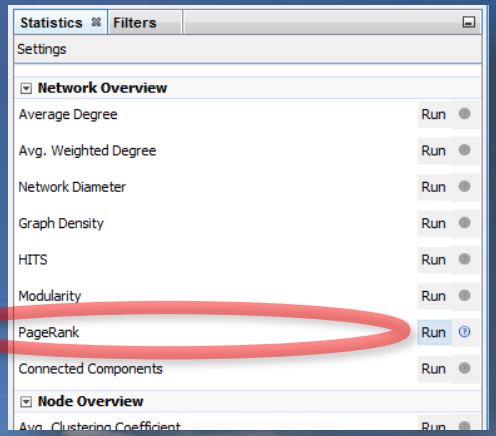
  - 1) Point to (Driver tab)
    httpS://dbpedia.org/sparql/
  - 2) Make a
    SPARQL **CONSTRUCT** query



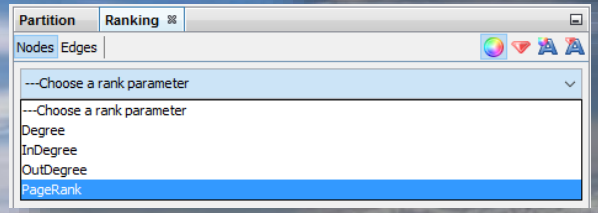
Gephi

- Importing data from DBpedia
  - 3) Run!
    - You get 2138 nodes and 4428 edges.
  - 4) Adjust visualization parameters
    - Run Statistics → PageRank
      - Default parameters



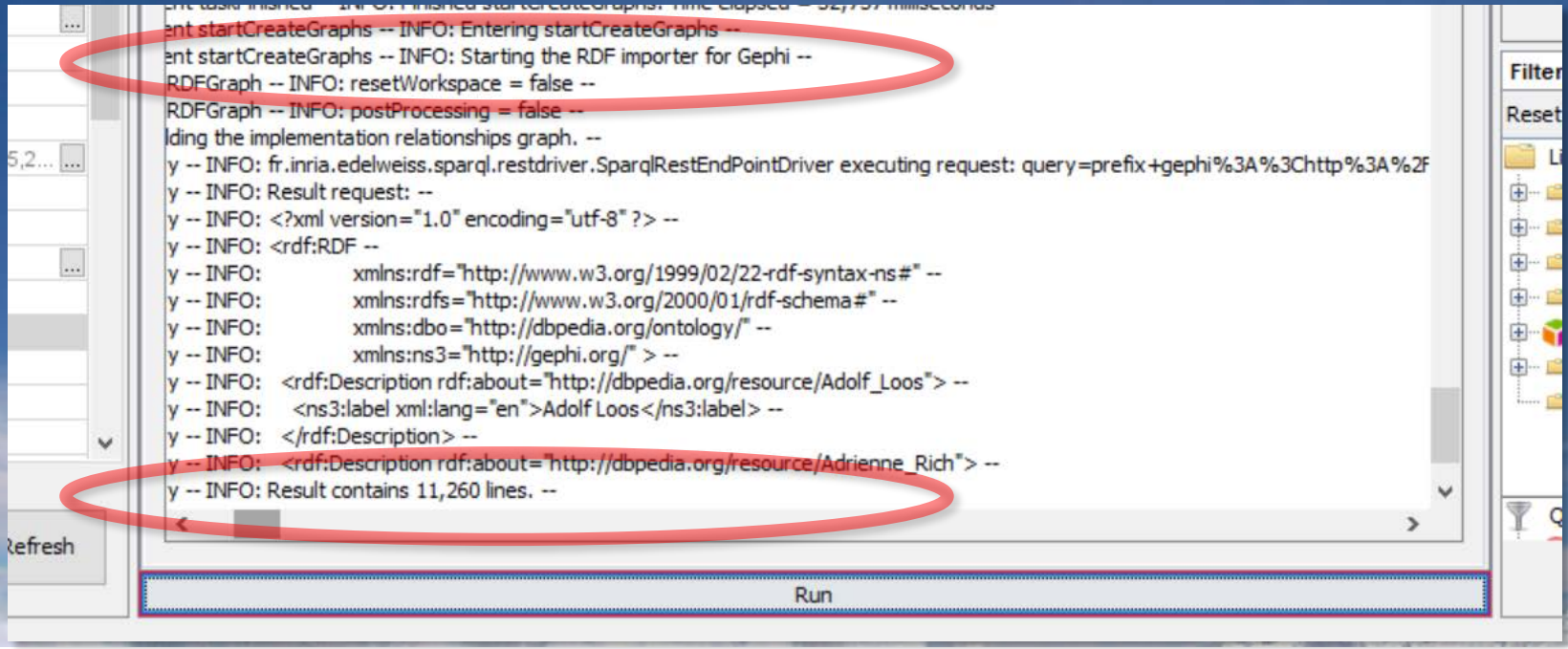


- Select Appearance → Nodes →Ranking
  Rank parameter = PageRank → Apply



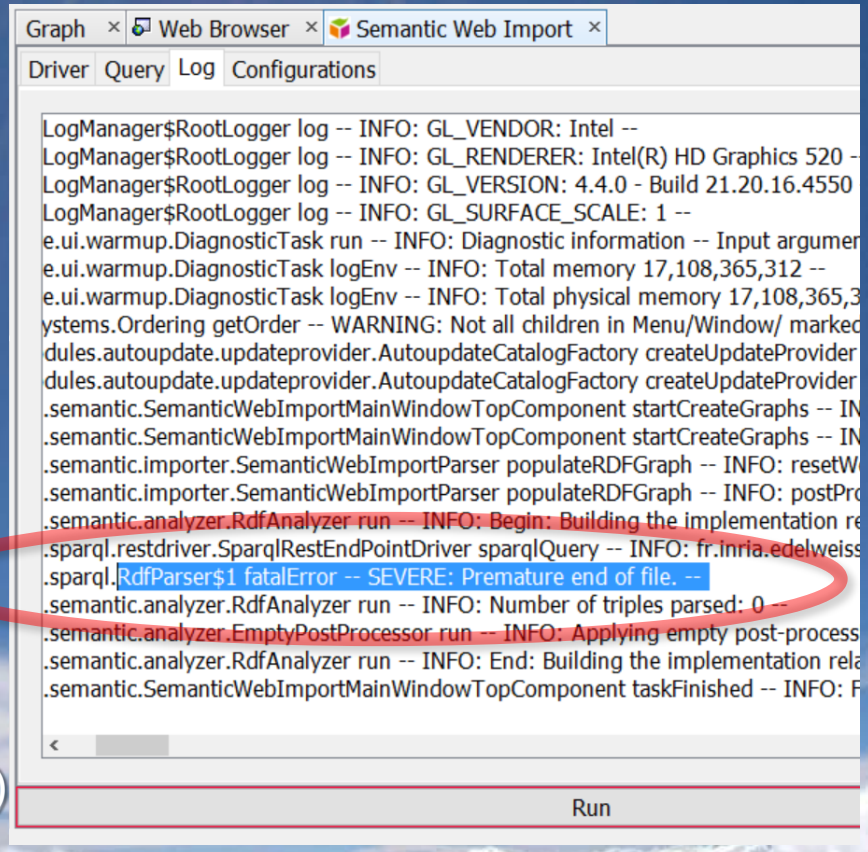
Gephi

- You should get something like this in the “log” tab



Gephi

- If you cannot make it run
  - See Log tab
    - “fatalError -- SEVERE:Premature end of file. --”



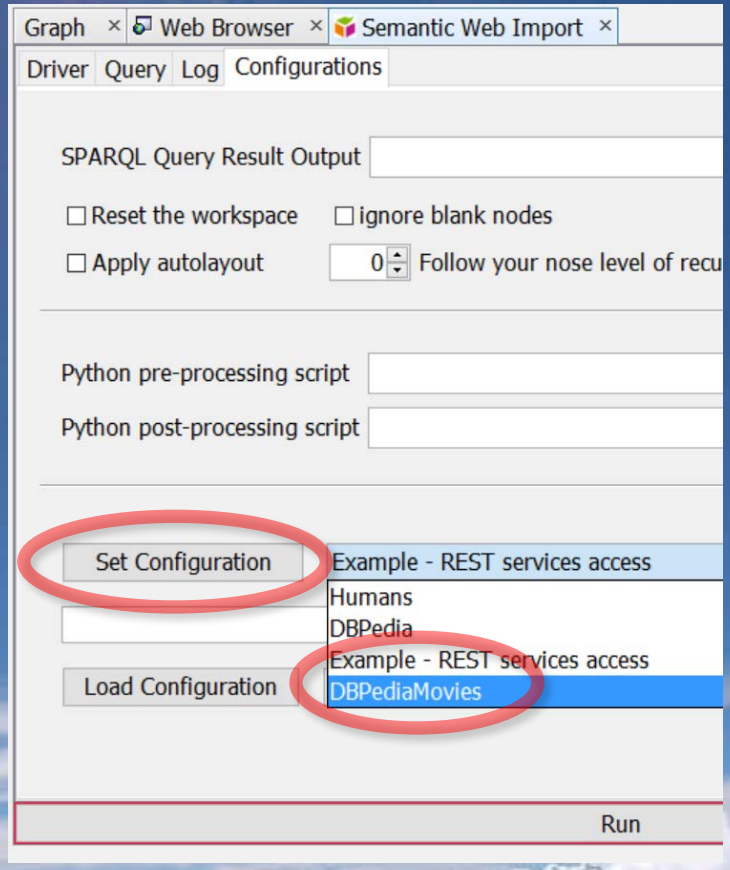
May be you forgot the trailing / in the URL of the SPARQL EP

Look for invalid chars in the query (e.g. tabs)

May be you have to update the plugins

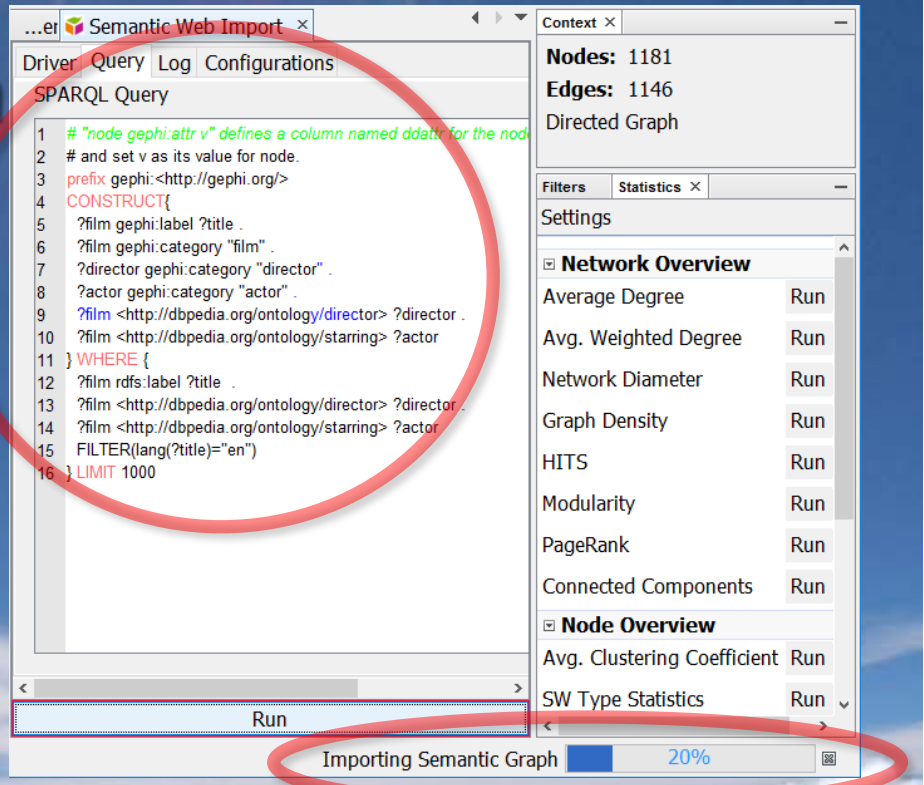
Gephi

- If you cannot make it run
  - Use a “pre configuration”
    - Select DBPediaMovies
    - Press button “Set configuration” (you can see the CONSTRUCT query in the “Query” tab)
    - Change http → https
    - Check the trailing /



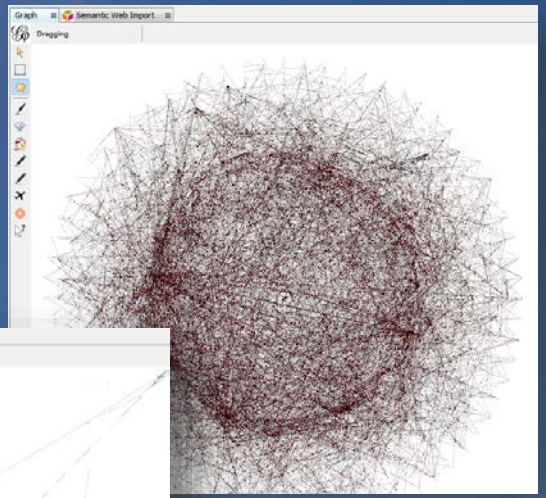
Gephi

- If you cannot make it run
  - Use a “pre configuration”
    - Select DBPediaMovies
    - Press button “Set configuration” (you can see the CONSTRUCT query in the “Query” tab)
    - Change http → https
    - Check the trailing /

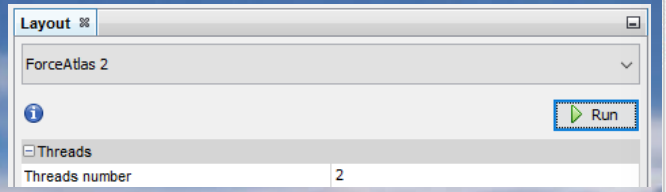


Gephi

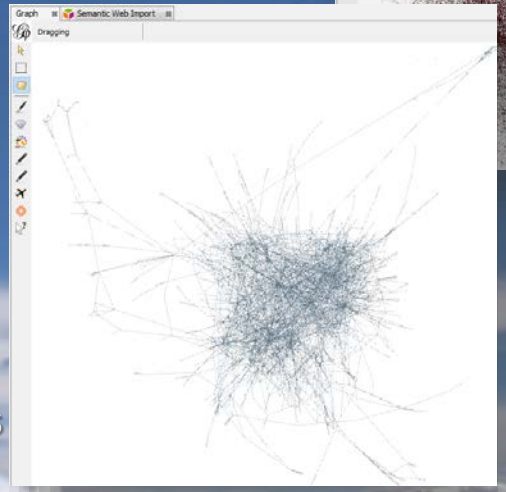
- Importing data from DBpedia
  - Check that you have something like this (Show/Hide the Graph tab with Window → Graph)



  - Layout → Force Atlas 2 → Run (~10 seconds)



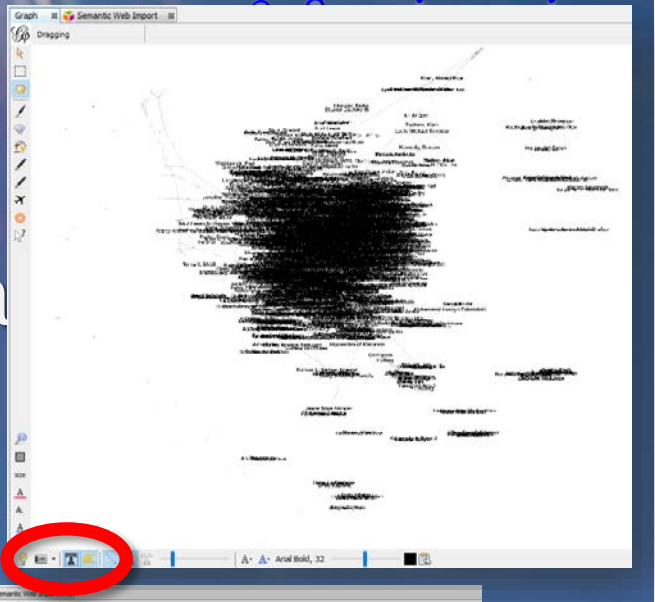
  - You should get some thing like this



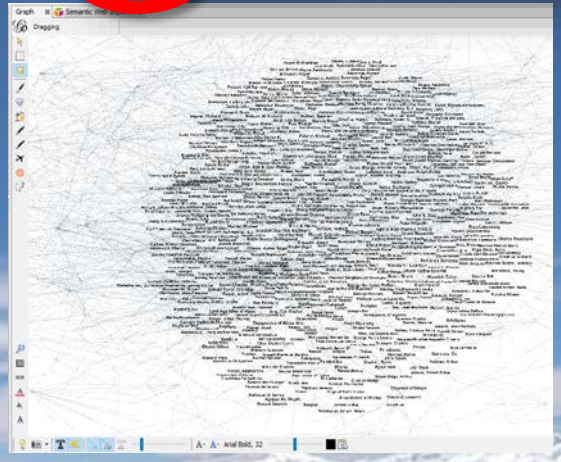
#VizLinkedData

Gephi

- Importing data from DBpedia
  - Show node labels
    (Clic icon  in the bottom of the Graph tab)

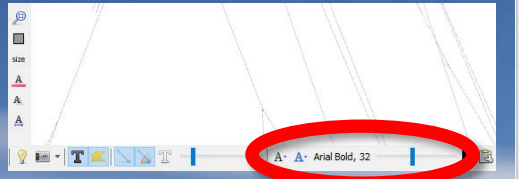


- Layout → Expansion → Run three times
- Layout → Label Adjust → Run ~10 seconds
- You should get some thing like this

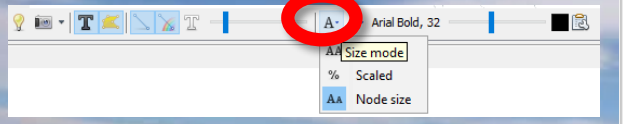


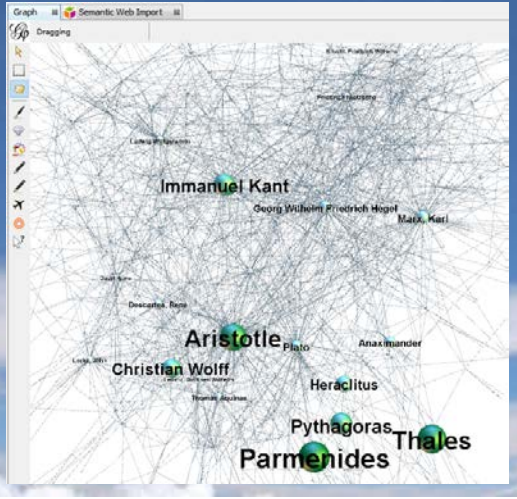
Gephi

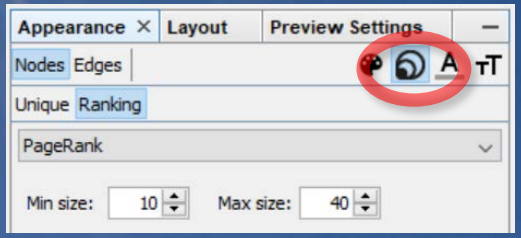
- Importing data from DBpedia
  - Node size proportional to ranking (Ranking→Nodes→ clic icon → set min & max →Apply)
  - Node label size proportional to ranking (Ranking→Nodes→clic icon → set min & max →Apply)
  - You should get some thing like this
    - Play with the font size slider

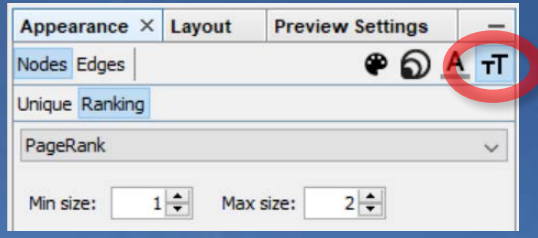


    - And the size mode



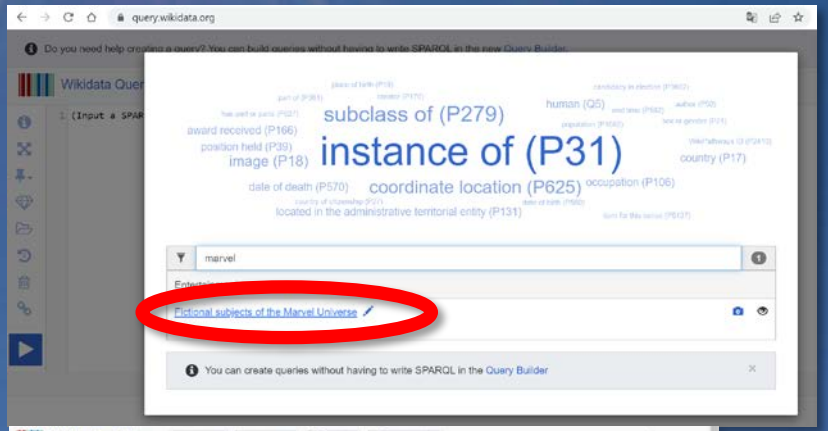


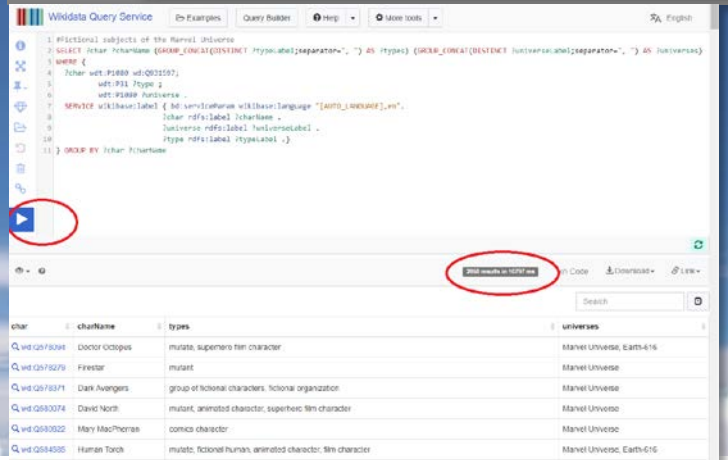




Gephi

- Importing data from Wikidata
  1) Use the Wikidata Query Service (SPARQL queries)
     - Select some example
     - Run the query



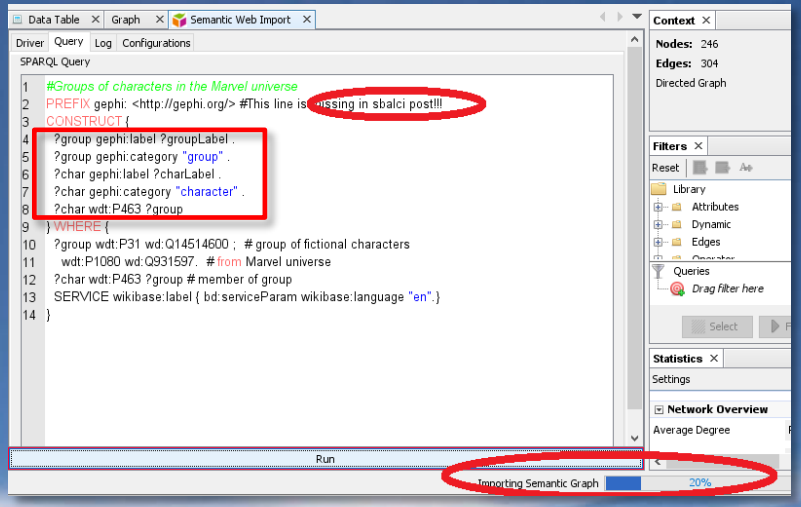


Gephi

- Importing data from Wikidata

Based on: sbalci repo at github

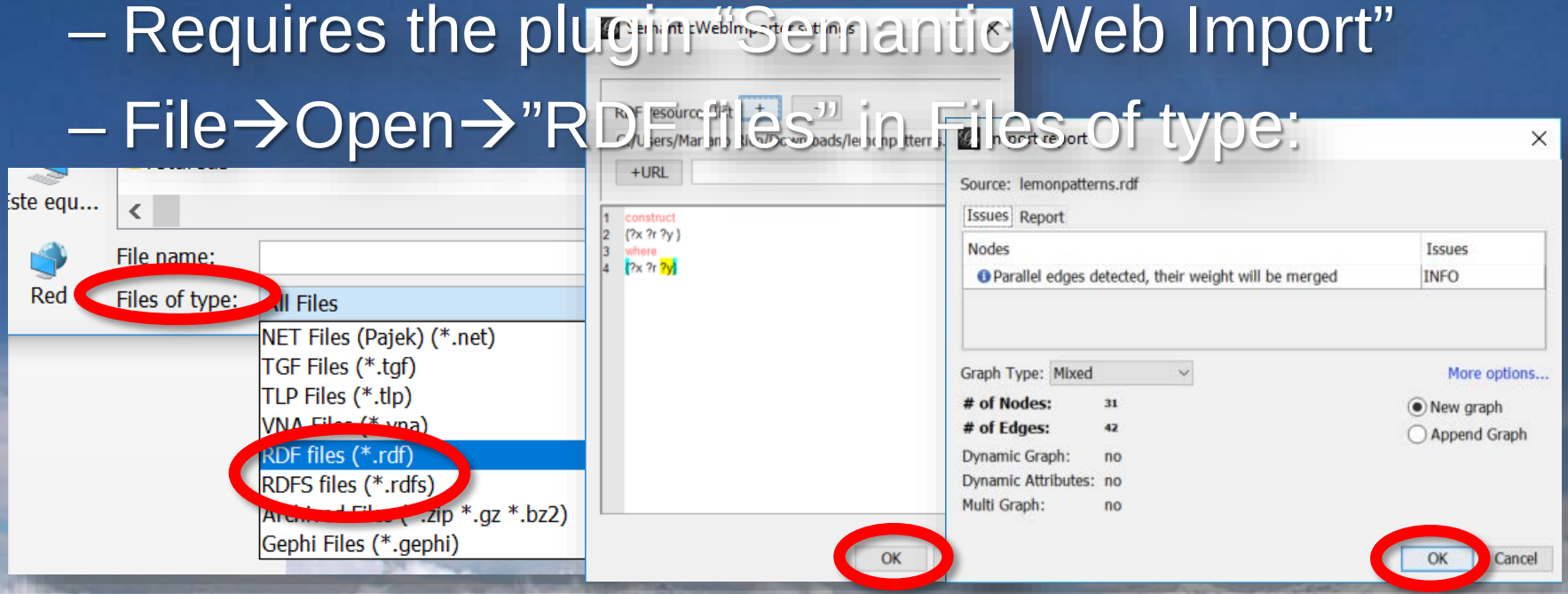
2) Create a SPARQL query with CONSTRUCT to create gephi data
   Notice: many prefixes known by the Wikidata EP



3) Point to the Wikidata endpoint `https://query.wikidata.org/sparql/`

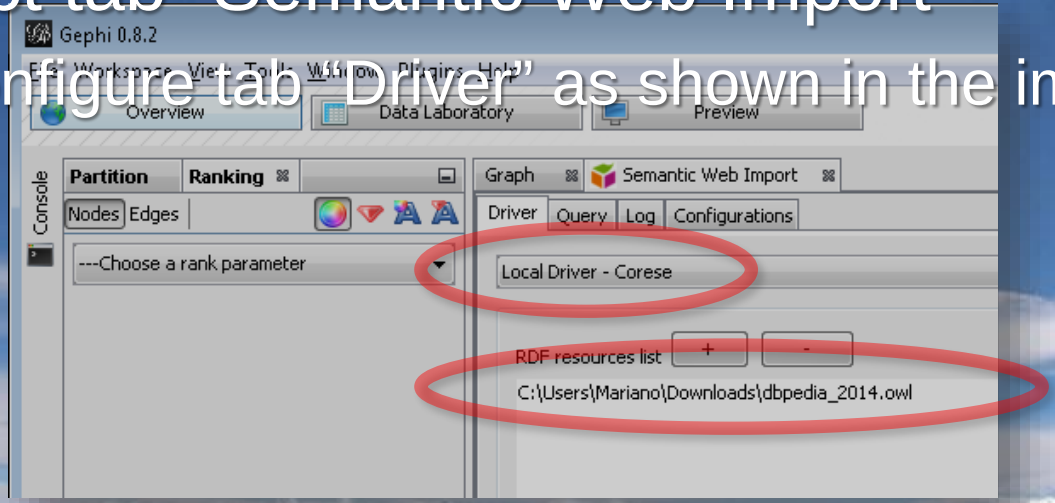
Gephi

- Gephi can import RDF (not .owl, see later)
  - Requires the plugin "Semantic Web Import"
  - File→Open→"RDF files" in Files of type:



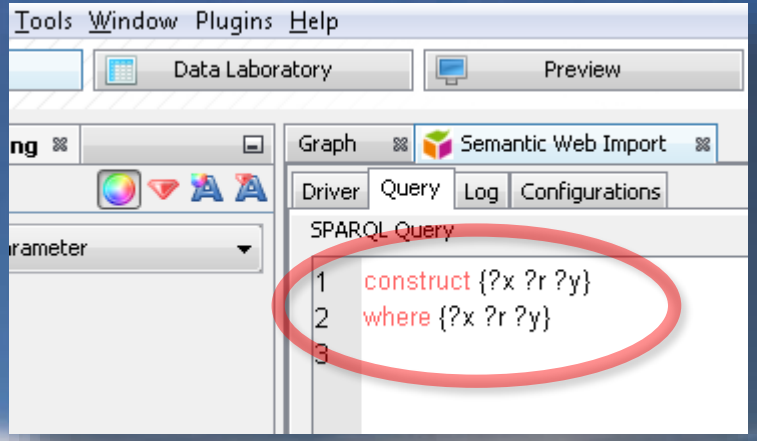
Gephi

- Other example: DBpedia ontology 2014
  - Load local file (dbpedia_2014.owl)
  - Select tab “Semantic Web Import”
    - Configure tab “Driver” as shown in the image



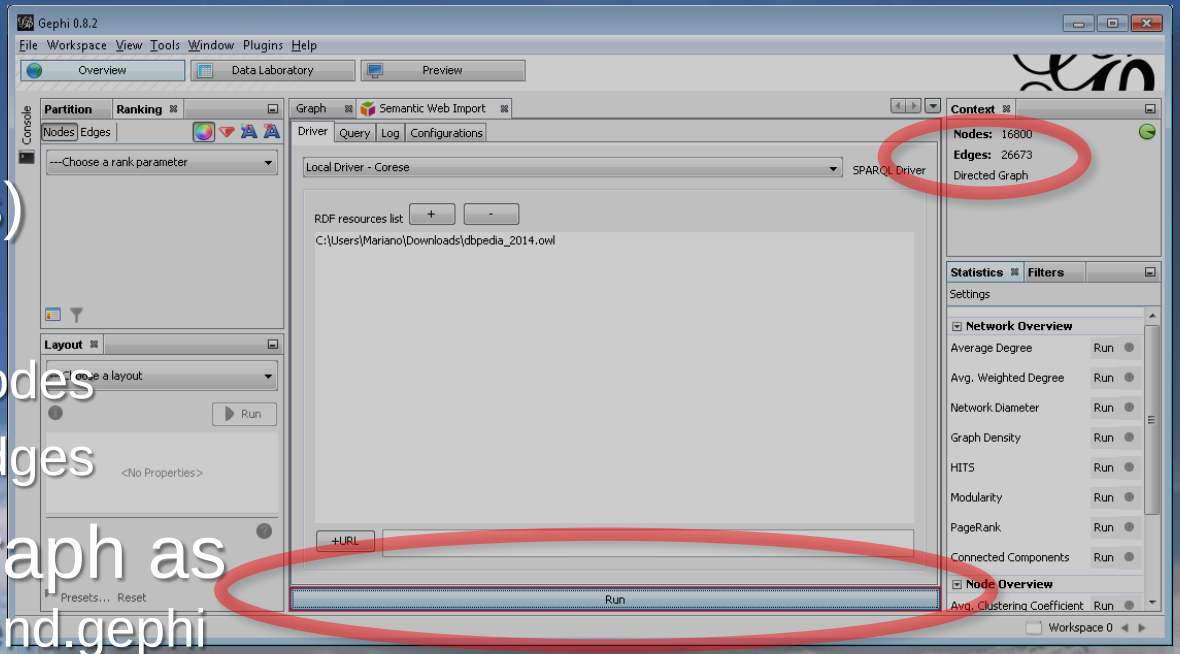
Gephi

- Load local file (dbpedia_2014.owl)
  2) Click tab “Query”
     - Remove the triples limit (100 by default)



Gephi

- Load local file (dbpedia_2014.owl)
  3) Run
     - Takes time (a few mins)
     - You get
       - 16,800 nodes
       - 26,673 edges
  4) Save the graph as dbpedia_2014.end.gephi



Gephi

- Load local file (dbpedia_2014.owl)

  Result after

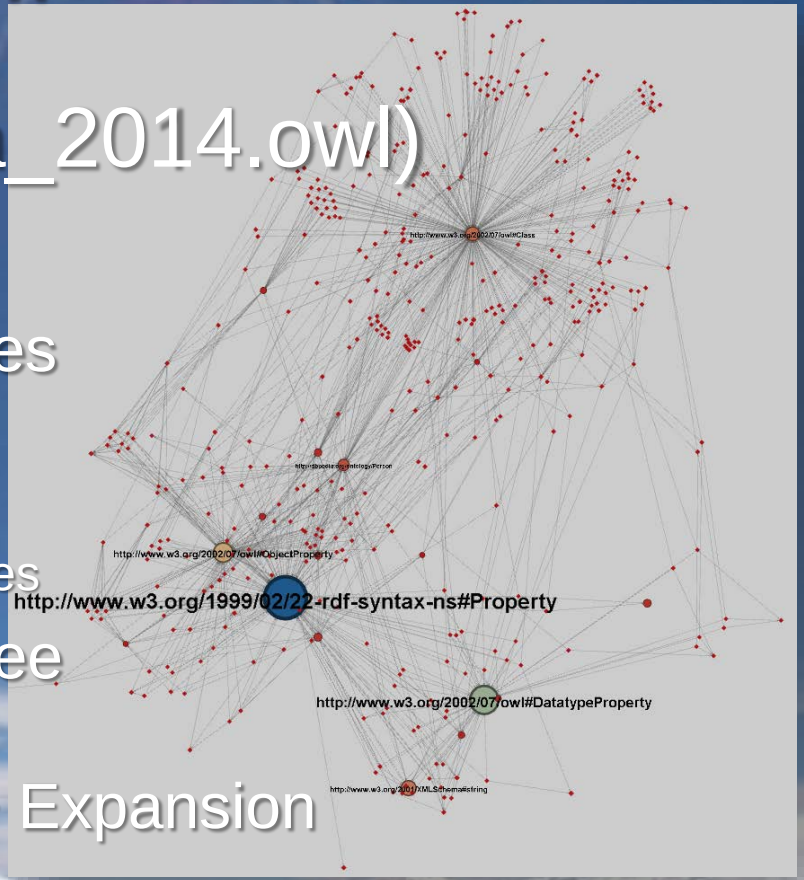
  – Filter nodes. “Remove” nodes with degree < 10

    – Makes visible only

      2.76% nodes and 4.5 edges

  – Node size and color ~ Degree
    (min size = 5 , max size = 50)

  – Fuchterman + Force atlas + Expansion



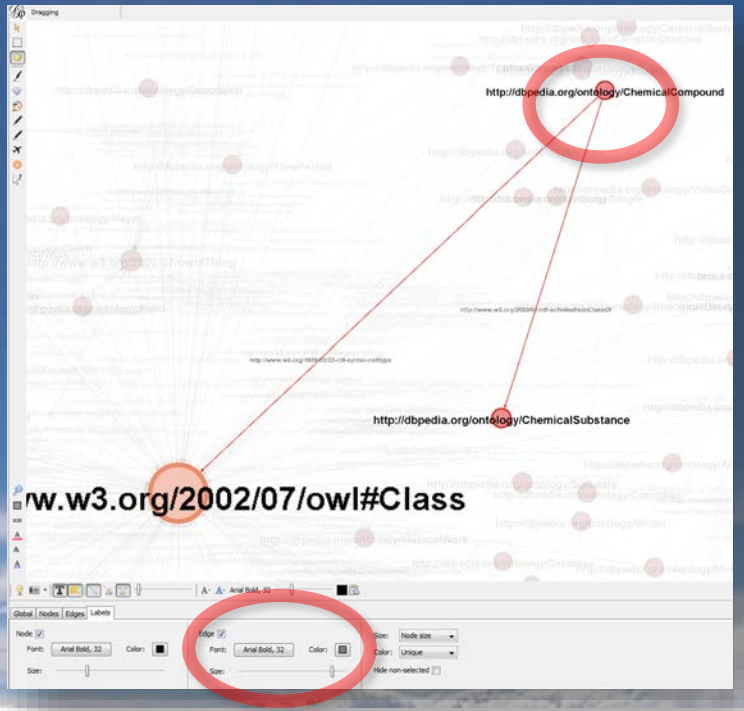
Gephi

- Load local file (dbpedia_2014.owl)

  Result analysis

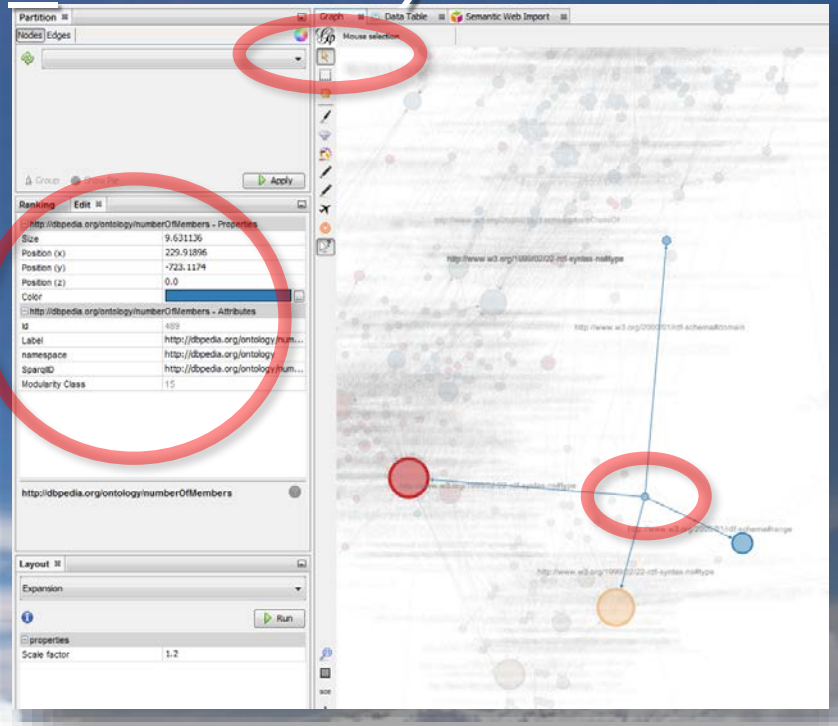
  – Zoom in into classes

    - Show edge labels
    - Hover a class (ChemicalCompound in the figure)



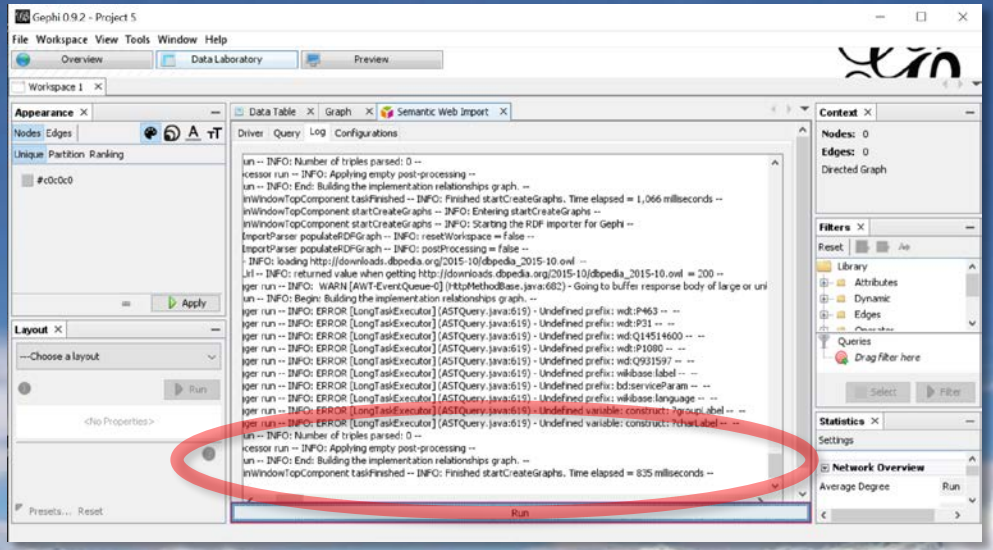
Gephi

- Load local file (dbpedia_2014.owl)
  - Result analysis
    - Use the selection tool 
      - Click on a node to see its properties in a tab (“Edit tab”)
      - Hide node labels
      - Increase edges’ label size
      - Now you see the node`s name in the “Edit tab” and its edges labels



Gephi

- Load online file
  - e.g. http://downloads.dbpedia.org/2015-10/dbpedia_2015-10.owl
- Execution error
  - At least on Windows with Gephi 0.9.2 ☹



Gephi

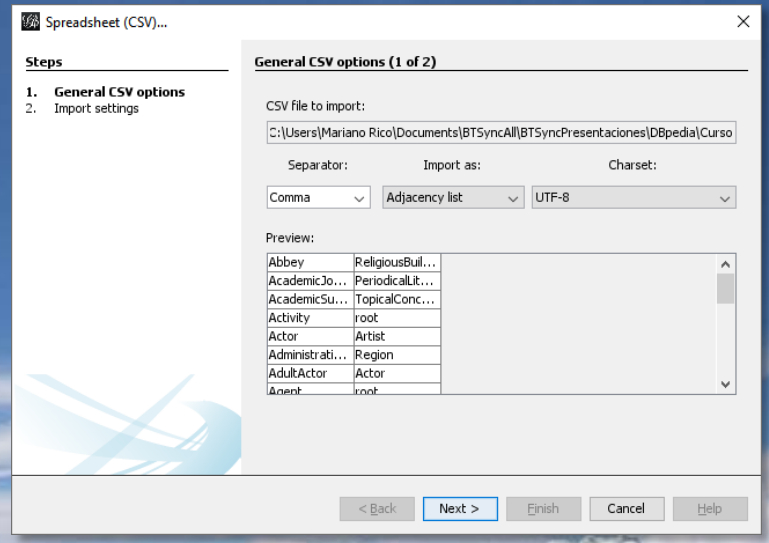
- Load local owl file
  - e.g. http://downloads.dbpedia.org/2016-10/dbpedia_2016-10.owl
- Goes fine
  - At least on Windows ☺
    - although it takes ~10 min to load ☹

Gephi. Load a csv file (1/3)

- e.g. DBpedia classes
  (with format `subclass,class`)
  - Use File→Open
  - Select the csv file
  - (opens a wizard)

DBclasses.csv

```
1  Abbey,ReligiousBuilding
2  AcademicJournal,PeriodicalLiterature
3  AcademicSubject,TopicalConcept
4  Activity,root
5  Actor,Artist
6  AdministrativeRegion,Region
7  AdultActor,Actor
8  Agent,root
9  Agglomeration,PopulatedPlace
10 Aircraft,MeanOfTransportation
```

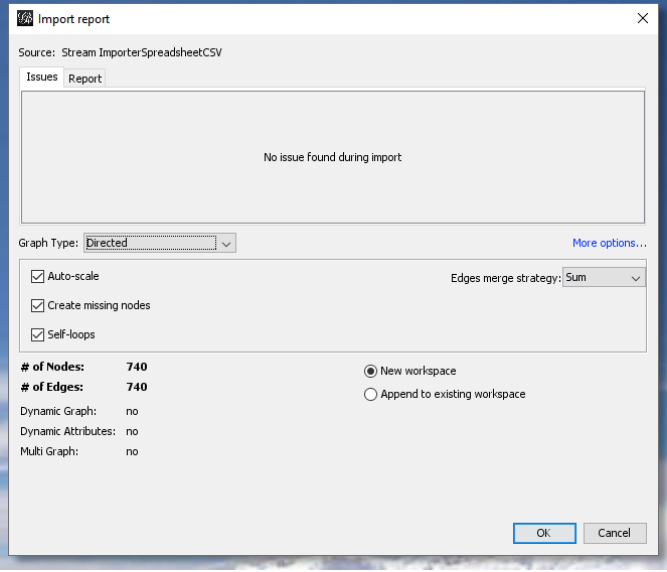


Gephi. Load a csv file (2/3)

- e.g. DBpedia classes
  (with format `subclass,class`)
  – Check the “import report”

DBclasses.csv

```
1  Abbey,ReligiousBuilding
2  AcademicJournal,PeriodicalLiterature
3  AcademicSubject,TopicalConcept
4  Activity,root
5  Actor,Artist
6  AdministrativeRegion,Region
7  AdultActor,Actor
8  Agent,root
9  Agglomeration,PopulatedPlace
10 Aircraft,MeanOfTransportation
```



Gephi. Load a csv file (3/3)

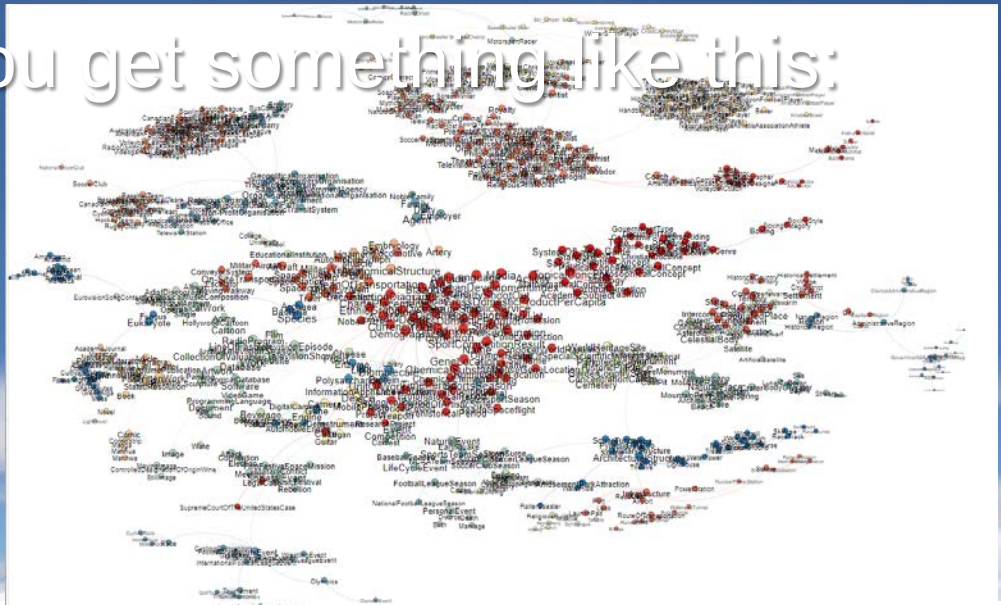
- e.g. DBpedia classes

  (with format `subclass,class`)

  – After some work you get something like this:

DBclasses.csv

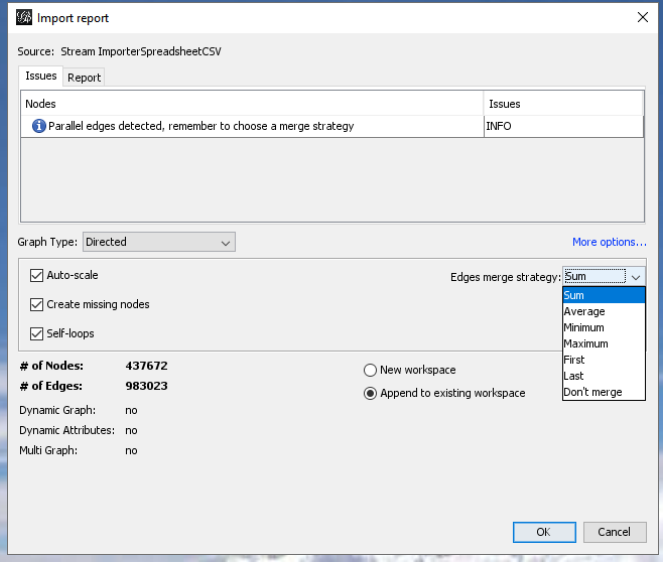
```
1  Abbey,ReligiousBuilding
2  AcademicJournal,PeriodicalLiterature
3  AcademicSubject,TopicalConcept
4  Activity,root
5  Actor,Artist
6  AdministrativeRegion,Region
7  AdultActor,Actor
8  Agent,root
9  Agglomeration,PopulatedPlace
10 Aircraft,MeanOfTransportation
```



Gephi. Load a csv file

- e.g. DBpedia categories (with format `subcat;cat`)
  - Graph10^3 bigger than classes graph
    - Warning!!
      - Parallel edges detected!!

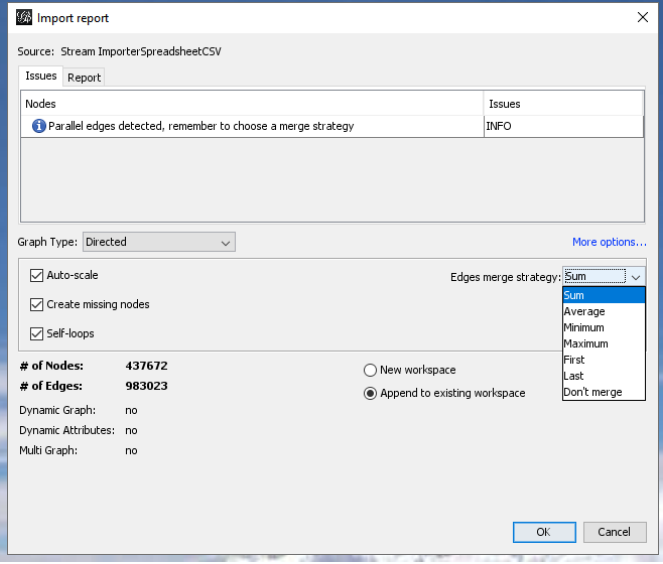
```
'N_Sync;Categorías_de_grupos_musicales_de_Estados_Unidos
'Ndrangheta;Historia_de_Calabria
'Ndrangheta;Organizaciones_delictivas_de_Italia
'Ndrangheta;Sociedades_secretas_criminales
(G)I-dle;Categorías_de_grupos_musicales_de_Corea_del_Sur
(G)I-dle;Grupos_de_pop_de_Corea_del_Sur
.hack;Categorías_de_series_de_anime_y_manga
.hack;Franquicias_de_videojuegos
.hack;Realidad_simulada_en_la_ficción
.hack;Realidad_virtual_en_ficción
.hack;Series_de_anime_de_Bandai_Visual
1._Bundesliga;Campeonatos_de_fútbol_entre_clubes_de_Alemania
```



Gephi. Load a csv file

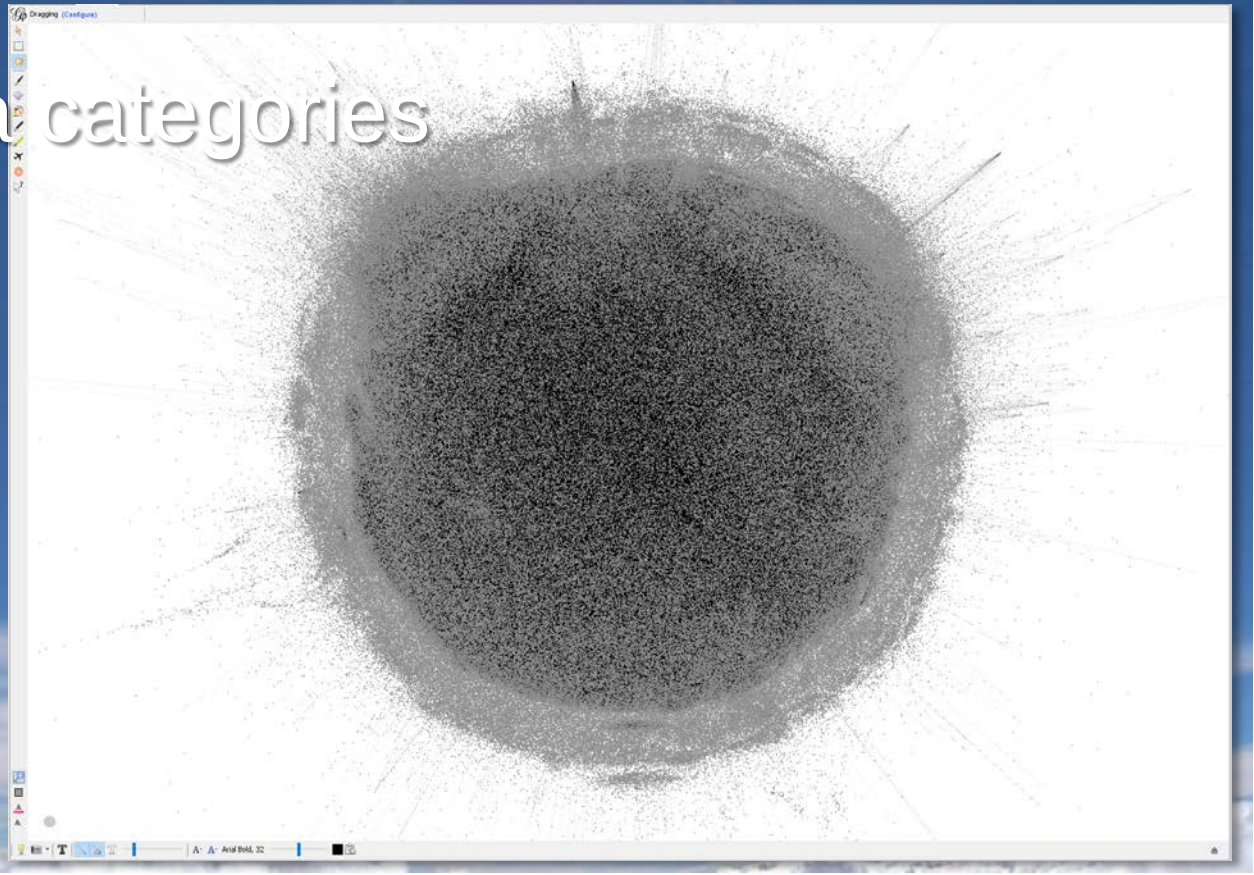
- e.g. DBpedia categories (with format `subcat;cat`)
  - Graph10^3 bigger than classes graph
    - Warning!!
      - Parallel edges detected!!

```
'N_Sync;Categorías_de_grupos_musicales_de_Estados_Unidos
'Ndrangheta;Historia_de_Calabria
'Ndrangheta;Organizaciones_delictivas_de_Italia
'Ndrangheta;Sociedades_secretas_criminales
(G)I-dle;Categorías_de_grupos_musicales_de_Corea_del_Sur
(G)I-dle;Grupos_de_pop_de_Corea_del_Sur
.hack;Categorías_de_series_de_anime_y_manga
.hack;Franquicias_de_videojuegos
.hack;Realidad_simulada_en_la_ficción
.hack;Realidad_virtual_en_ficción
.hack;Series_de_anime_de_Bandai_Visual
1._Bundesliga;Campeonatos_de_fútbol_entre_clubes_de_Alemania
```



Gephi. Load a csv file

- e.g. DBpedia categories

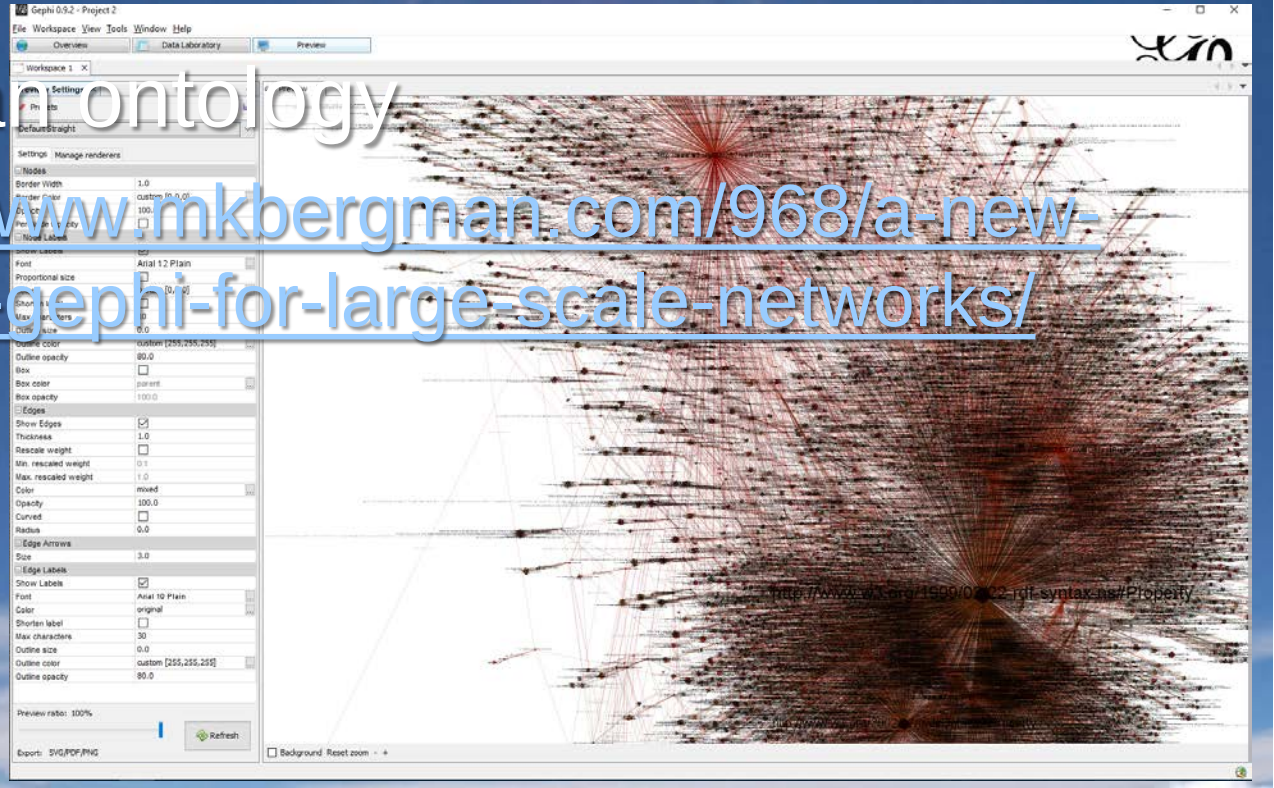


Gephi. Load a csv file

- e.g. DBpedia categories

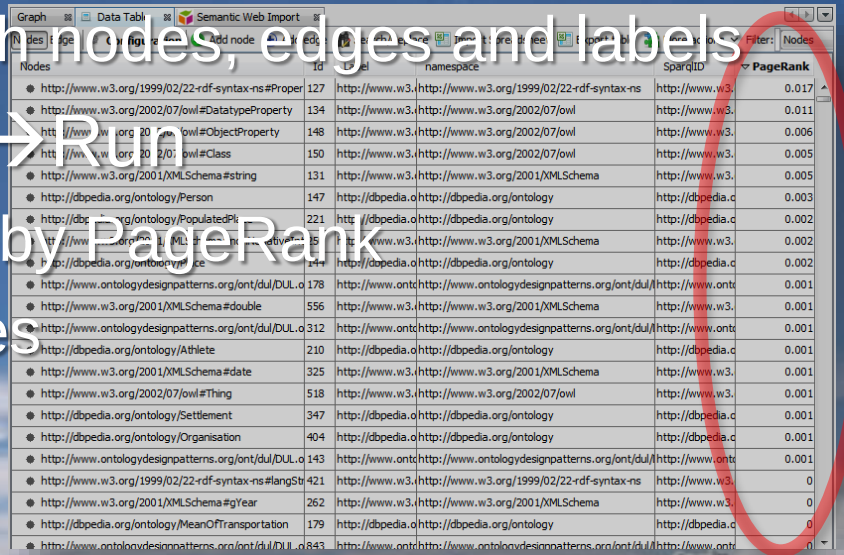
Gephi

- Displaying an ontology
  - See http://www.mkbergman.com/968/a-new-best-friend-gephi-for-large-scale-networks/



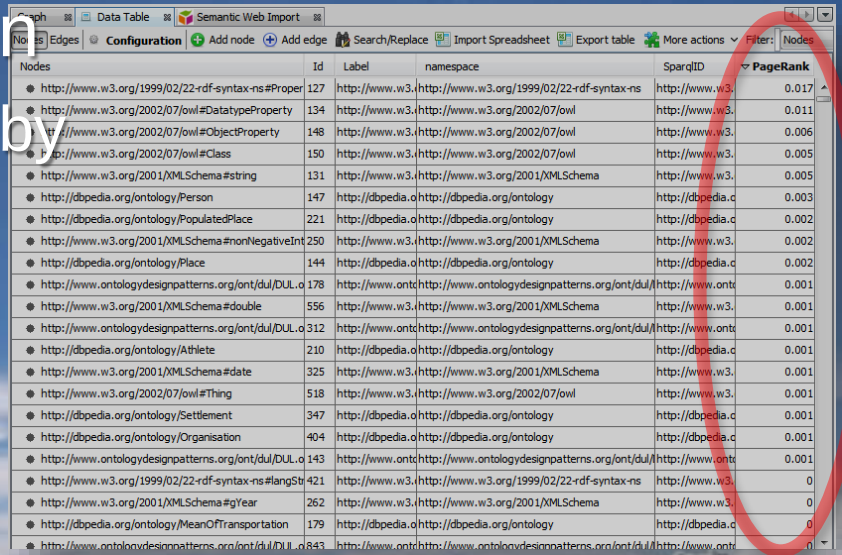
Gephi

- The DBpedia Ontology use case
  - Load the file `dbpedia_2014.init.gephi`
    - It is a directed graph with nodes, edges and labels
  - Statistics → PageRank → Run
    - See the data table, sort by PageRank
    - Notice that only 19 nodes have a value != 0 →
      Not a valid measure



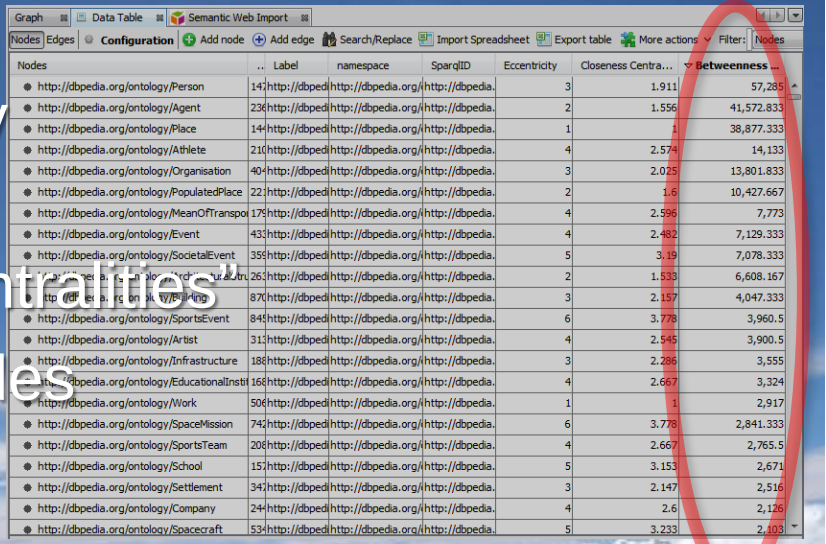
Gephi

- The DBpedia Ontology use case
  - Statistics →Edge Overview → Avg. Path Length→ Run
    - See the data table, sort by Betweenness Centrality
    - Tic normalize
    - Notice that ALL nodes have a value == 0 → **Not a valid measure**



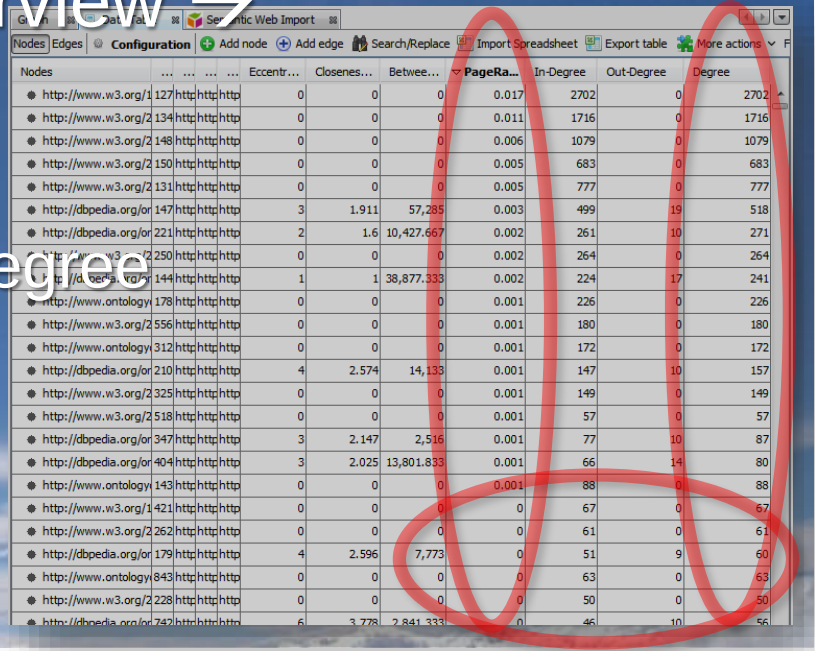
Gephi

- The DBpedia Ontology use case
  - Statistics →Edge Overview → Avg. Path Length→ Run
    - See the data table, sort by Betweenness Centrality
    - Do NOT tic “normalize centralities”
    - Notice that now, more nodes have a value != 0 → **Do NOT normalize!!**



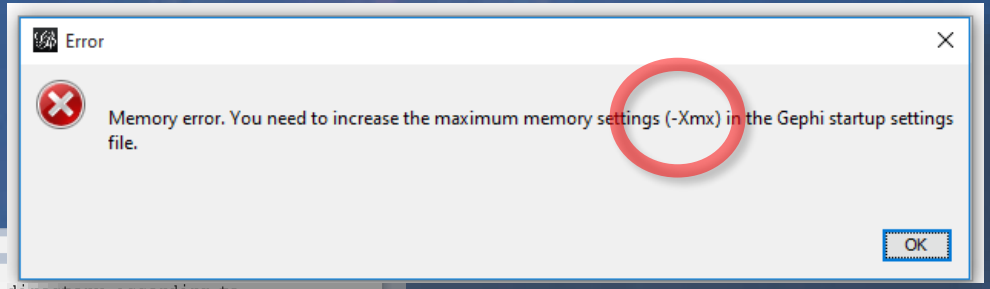
Gephi

- The DBpedia Ontology use case
  - Statistics →Network Overview → Average Degree→ Run
    - See the data table, with: In-Degree, Out-Degree, Degree
    - Degree runs parallel to PageRank
      - Notice Degree has more nodes with value != 0 ☺



Gephi limitations

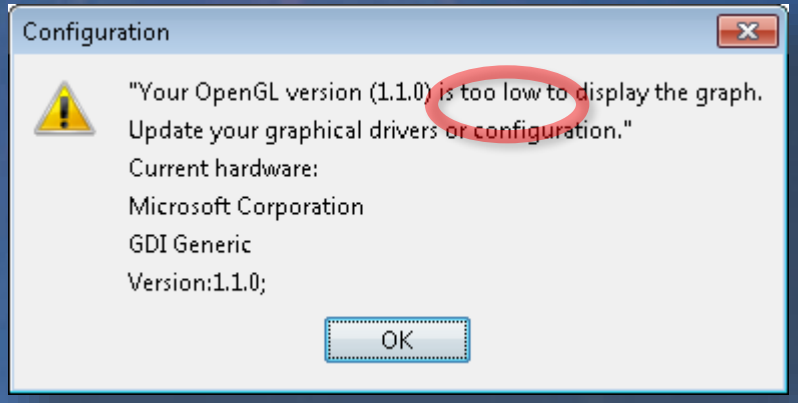
- Out of memory
  - Edit {gephiDir}/etc/gephi.conf




```
# ${HOME} will be replaced by user home directory according to
platform
default_userdir="${HOME}/.${APPNAME}/0.8.2/dev"
default_mac_userdir="${HOME}/Library/Application Support/
${APPNAME}/0.8.2/dev"

# options used by the launcher by default, can be overridden
by explicit
# command line switches
default_options="--branding gephi -J-Xms64m -J-Xmx512m -J-
Xverify:none -J-Dsun.java2d.noddraw=true -J-
Dsun.awt.noerasebackground=true -J-
Dnetbeans.indexing.noFileRefresh=true -J-
Dplugin.manager.check.interval=EVERY_DAY -J-Duser.language=en"
# for development purposes you may wish to append: -J-
Dnetbeans.logger.console=true -J-ea

# default location of JDK/JRE, can be overridden by using --
jdkhome <dir> switch
jdkhome="C:\Program Files\Java\jdk1.7.0_79"

# clusters' paths separated by path.separator (semicolon on
Windows, colon on Unices)
#extra_clusters=
```

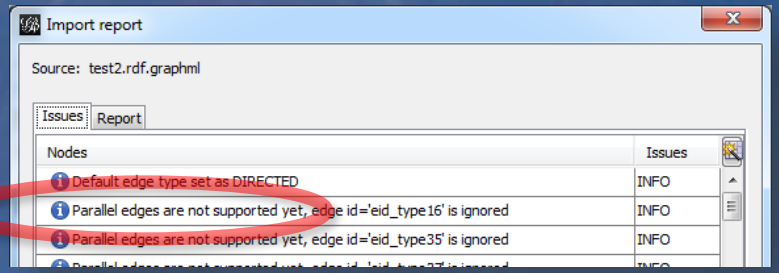
Gephi limitations

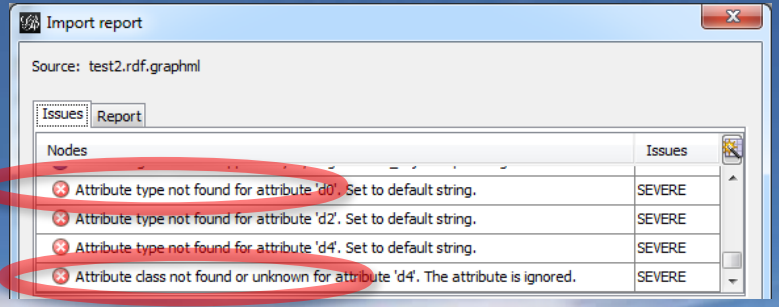
- OpenGL
  - Update your graph drivers
    - For Intel go here



Gephi limitations

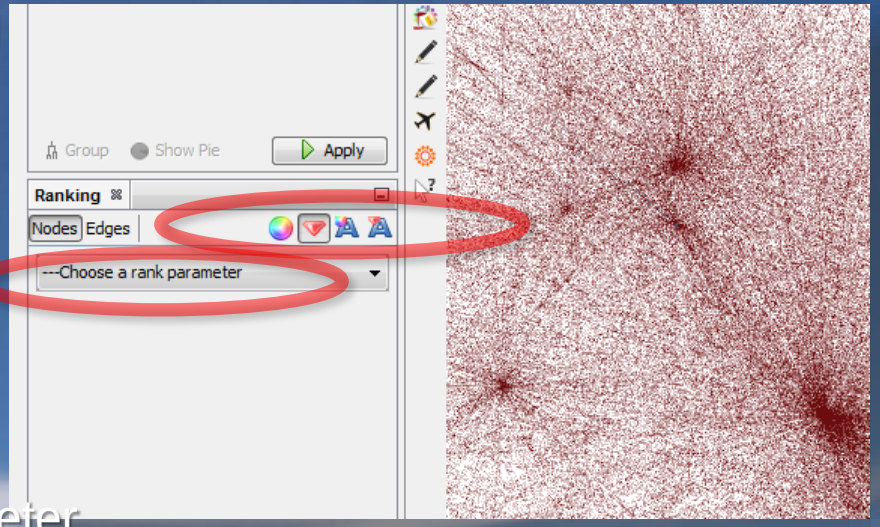
- Logical
  - Not supported (yet)
    - Parallel edges
      (neither direct and reverse)





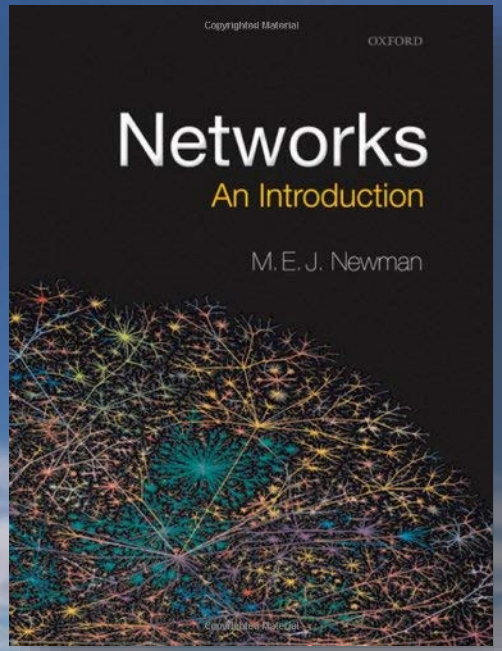
Gephi limitations

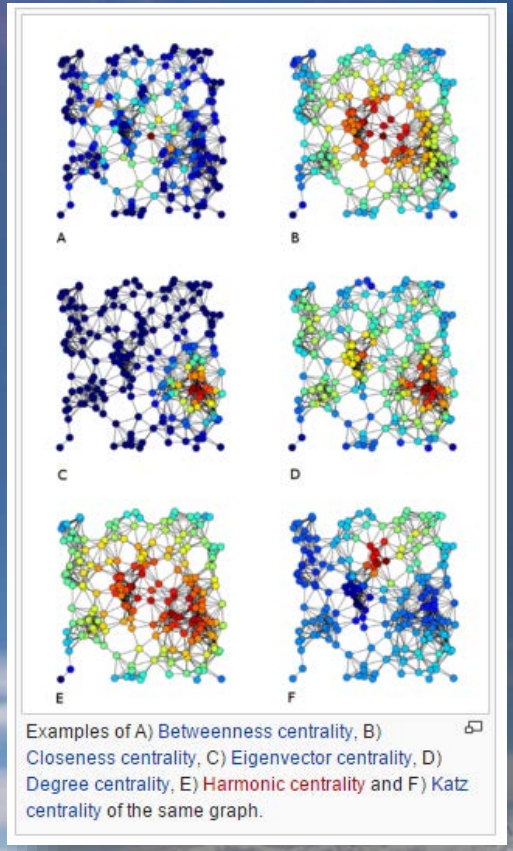
- Usability
  - UI selectors are confusing
    - Always show the last used option
    - They do not use the current parameter ☹
    - E.g. load a graph
      - In the “Ranking” tab you cannot know which parameter has been used for node size, color, etc.



More on graph theory

- Centrality
  - See on wikipedia
- Graphs
  - Newman's book, 2010.





Examples of A) Betweenness centrality, B) Closeness centrality, C) Eigenvector centrality, D) Degree centrality, E) Harmonic centrality and F) Katz centrality of the same graph.

More on graph theory

- Classic papers on Wikipedia category graph analysis (and its applications)
  - Identifying document topics using the Wikipedia category network (2009)
  - Decoding Wikipedia Categories for Knowledge Acquisition (2008)
  - Analysis of the Wikipedia Category Graph for NLP Applications (2007)

More on graph theory

- Classic papers on Wikipedia category graph analysis (and its applications)
  - Identifying document topics using the Wikipedia category network (2009)

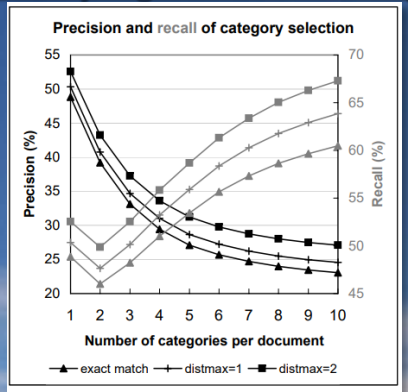


Figure 7. Black: percentage of Wikipedia categories which were correct in the top n categories. Gray: percentage of official Wikipedia categories among the top n categories. In both cases, n is shown on the x axis, values are averaged over processed documents.

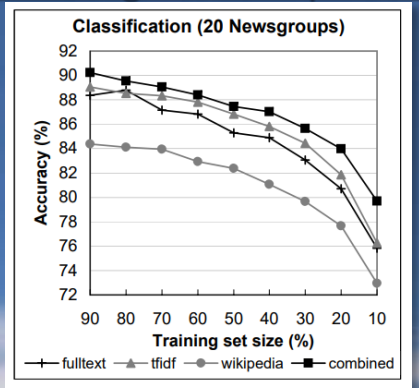


Figure 8. Classification accuracy in 20 Newsgroups at various training set sizes, when documents were represented by full text ("fulltext"), the 20 words with highest $tf \times idf$ ("tfidf"), top 20 categories ("wikipedia"), or combination of the latter two ("combined").

More on graph theory

- Classic papers on Wikipedia category graph analysis (and its applications)
  - Identifying document topics using the Wikipedia category network (2009)
  - Decoding Wikipedia Categories for Knowledge Acquisition (2008)

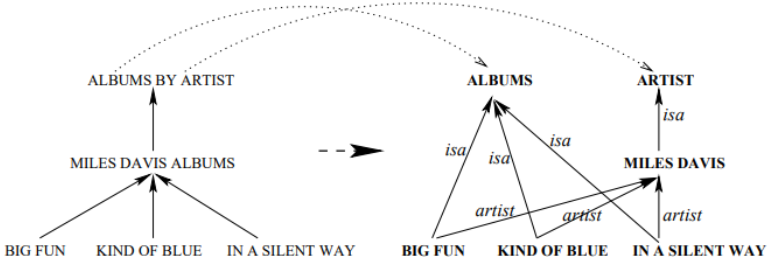


Figure 3: Relations inferred from "by" categories

Category type	# categories	# relations extracted	Evaluation		
			P	manual ∩	manual ∪
explicit relations	3,450	86,649			
caused_by, *based_in*, *written_by*, ...	2,152	43,938	-	94.37%	96.38%
member_of	1,298	42,711	24% (25)	95.56%	97.17%
partly explicit and implicit relation categories	98,855	9,751,748			
isa		3,400,243	44.57% (6,250)	76.4%	84%
spatial		3,201,125	39.69% (1,325)	87.09%	97.98%

Table 2: Extracted relations and evaluation results

More on graph theory

- Classic papers on Wikipedia category graph analysis (and its applications)
  - Identifying document topics using the Wikipedia category network (2009)
  - Decoding Wikipedia Categories for Knowledge Acquisition (2008)
  - Analysis of the Wikipedia Category Graph for NLP Applications (2007)

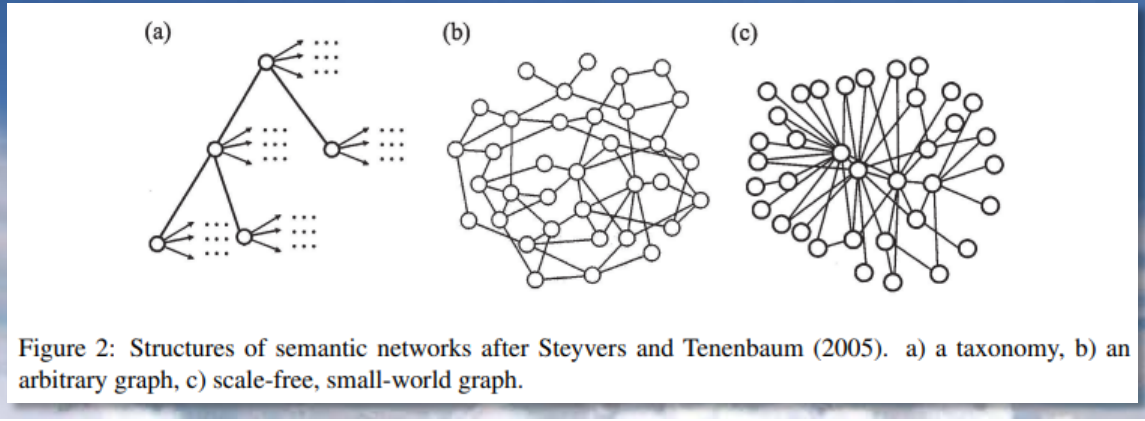


Figure 2: Structures of semantic networks after Steyvers and Tenenbaum (2005). a) a taxonomy, b) an arbitrary graph, c) scale-free, small-world graph.

More on graph theory

- Less classic papers on Wikipedia category graph analysis (and its applications)
  - Uncovering the Semantics of Wikipedia Categories (2019)

Table 2. Total number of axioms/assertions and precision scores, based on the crowd-sourced evaluation. Numbers in parentheses denote the *total* number of assertions generated (including those already existing in DBpedia), as well as the precision estimation of those total numbers. The latter were derived as a weighted average from the human annotations and the overall correctness of existing assertions in DBpedia according to [3].

Approach	Count	Precision [%]	Count	Precision [%]
	Relation axioms		Type axioms	
Cat2Ax	272,707	95.6	430,405	96.8
C-DF	143,850	83.6	28,247	92.0
Catriple	306,177	87.2	–	–
	Relation assertions		Type assertions	
Cat2Ax	4,424,785 (7,554,980)	87.2 (92.1)	3,342,057 (12,111,194)	90.8 (95.7)
C-DF	766,921 (2,856,592)	78.4 (93.4)	198,485 (2,352,474)	76.8 (97.1)
Catriple	6,260,972 (6,836,924)	74.4 (76.5)	–	–

More on graph theory

- Less classic papers on Wikipedia category graph analysis (and its applications)
  - Cat2Type: Wikipedia Category Embeddings for Entity Typing in Knowledge Graphs (2021)

Table 2: Results on DBpedia splits and FIGER

Models	DB1		DB2		DB3		FIGER	
	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$	$Ma-F_1$	$Mi-F_1$
CUTE [32]	0.679	0.702	0.681	0.713	0.685	0.717	0.743	0.782
MuLR [34]	0.748	0.771	0.757	0.784	0.752	0.775	0.776	0.812
FIGMENT [33]	0.740	0.766	0.738	0.765	0.745	0.769	0.785	0.819
APE [16]	0.758	0.784	0.761	0.785	0.760	0.782	0.722	0.756
HMGCN [17]	0.785	0.812	0.794	0.820	0.791	0.817	**0.789**	0.827
Cat2Type-node2vec	0.950	0.948	0.948	0.946	0.948	0.946	0.683	0.84
Cat2Type-word2vec	0.876	0.876	0.723	0.738	0.723	0.742	0.502	0.726
Cat2Type-GloVE	0.883	0.884	0.728	0.742	0.731	0.746	0.501	0.726
Cat2Type-Wikipedia2Vec	0.897	0.897	0.733	0.749	0.739	0.754	0.522	0.737
Cat2Type-BERT	**0.983**	**0.984**	**0.983**	**0.983**	**0.985**	**0.985**	0.764	**0.881**

More on Gephi Layouts

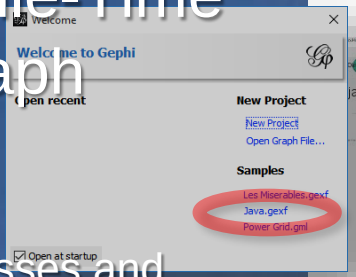
- Online tutorial
  - Gephi web site
- Book
  - Ken Cherven, *Mastering Gephi network visualization*. Packt Publishing, 2015.

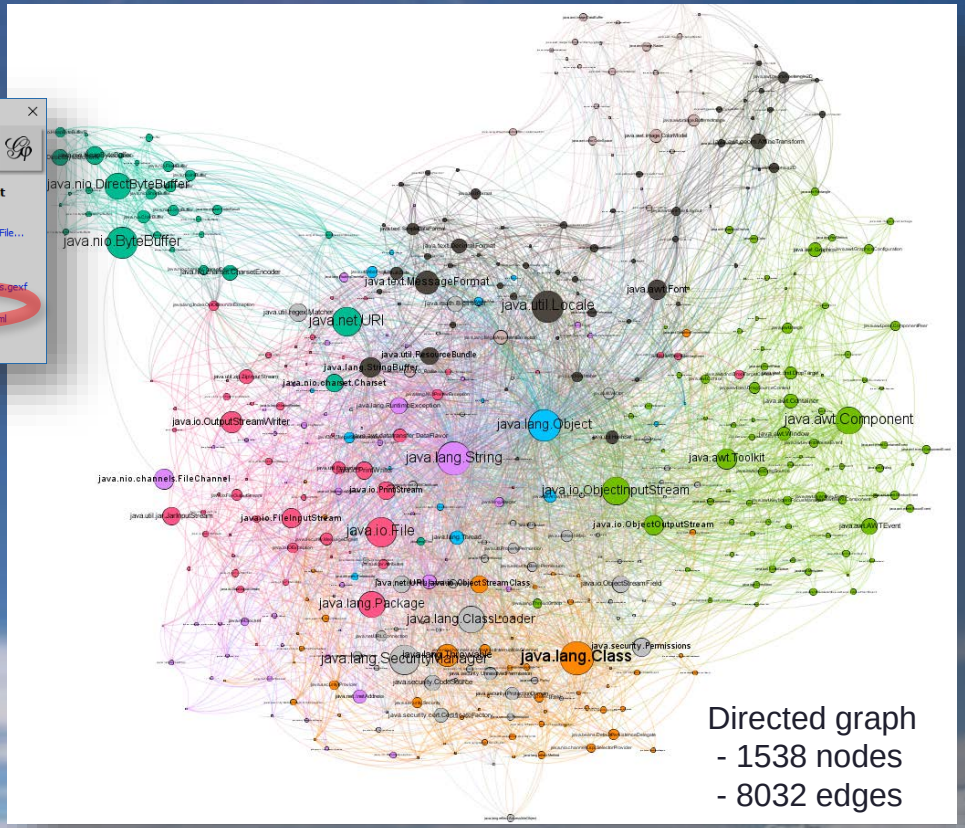
Selecting the Layout

Algorithm name	Type	Strengths	Weaknesses	When to use
DAG layout	Tree	Ordering hierarchical data	This is impractical for very large networks	This can be used in cases where you wish to see levels of data in a top to bottom order
Dual Circle layout	Circular	This has the ability to focus on a group of nodes within the larger network	This layout results in very large networks that might create viewing issues	This layout can be used in instances where a second circle is desirable to focus on a limited group of nodes
Force Atlas	Force-directed	This includes many options and has a high level of accuracy	This can be very slow and is not suited to large networks	This layout is useful for network analysis and discovery, and for measuring network behavior
Force Atlas 2	Force-directed	This is faster than original Force Atlas and handles very large networks	This suffers slightly on overall accuracy	This is used as a good tool for network analysis and discovery, and for detecting behavioral patterns

More examples

- The Java Compile-Time Dependency graph
  - Gephi dataset
    - Java.gexf
  - Nodes are Java classes and directed edges are compile-time dependencies between two classes.
  - The data provided contains the dependencies for all classes under the java.* packages for JDK 1.4.2
  - From 2006 International Symposium on Graph Drawing

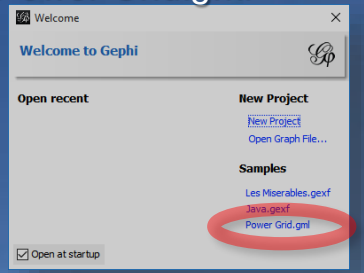




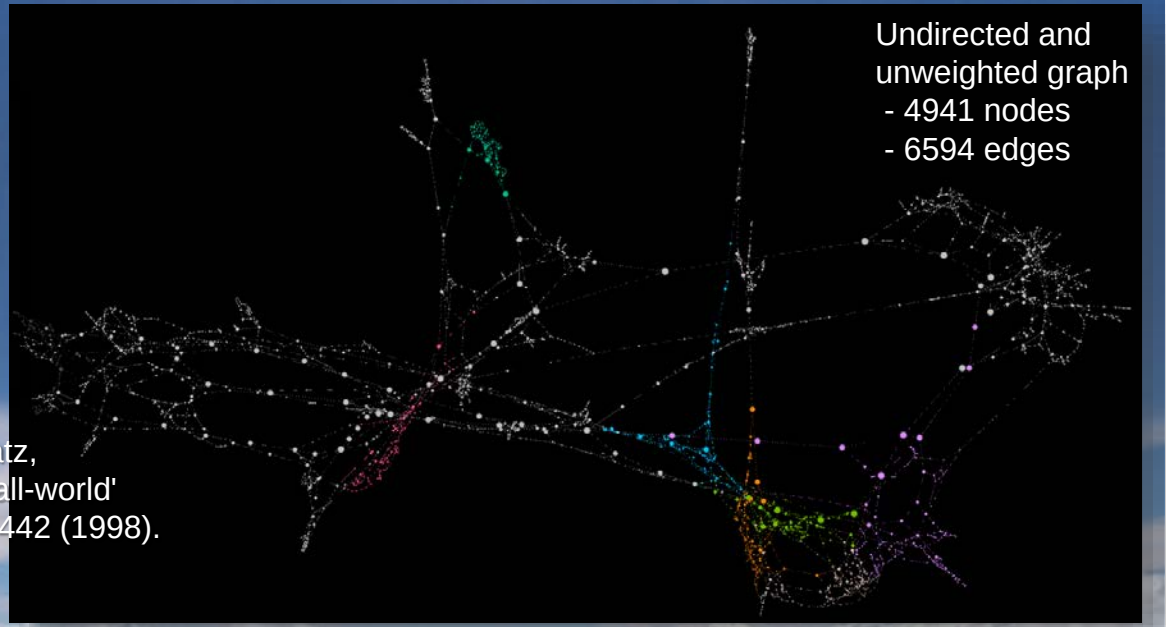
Directed graph
- 1538 nodes
- 8032 edges

More examples

- Topology of the Western States power grid in U.S.
  - Gephi dataset
    - Power Grid.gml



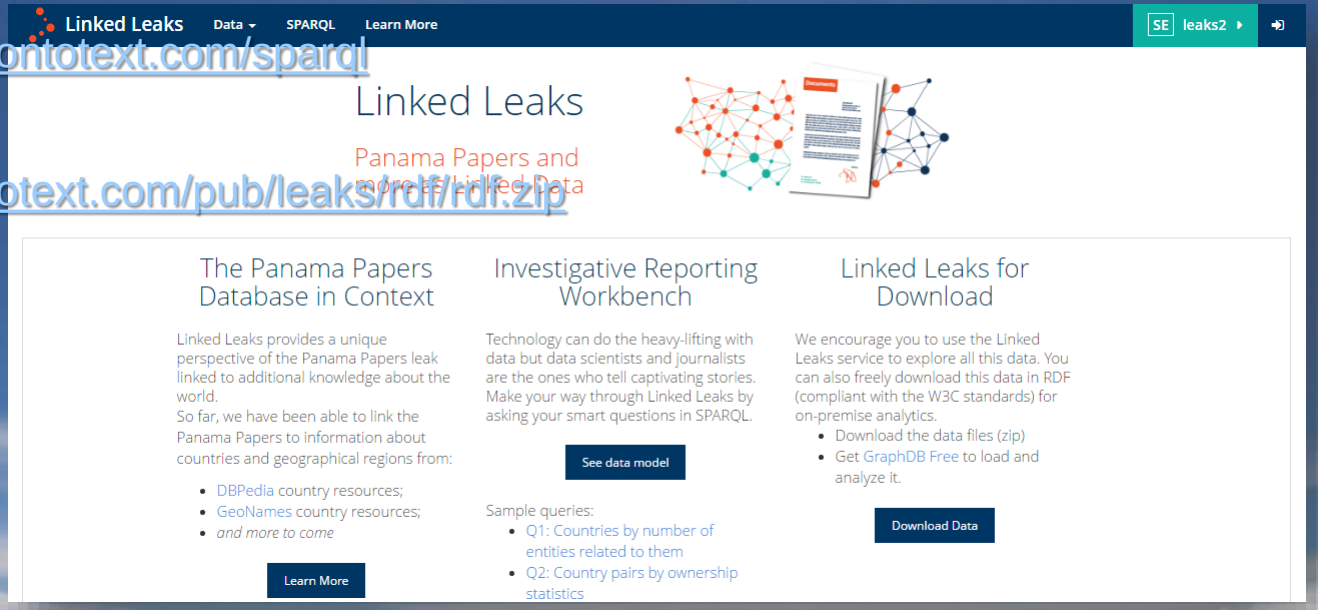
  - Original paper citation
    - D. J. Watts and S. H. Strogatz, "Collective dynamics of `small-world' networks", Nature 393, 440-442 (1998).



Undirected and unweighted graph
- 4941 nodes
- 6594 edges

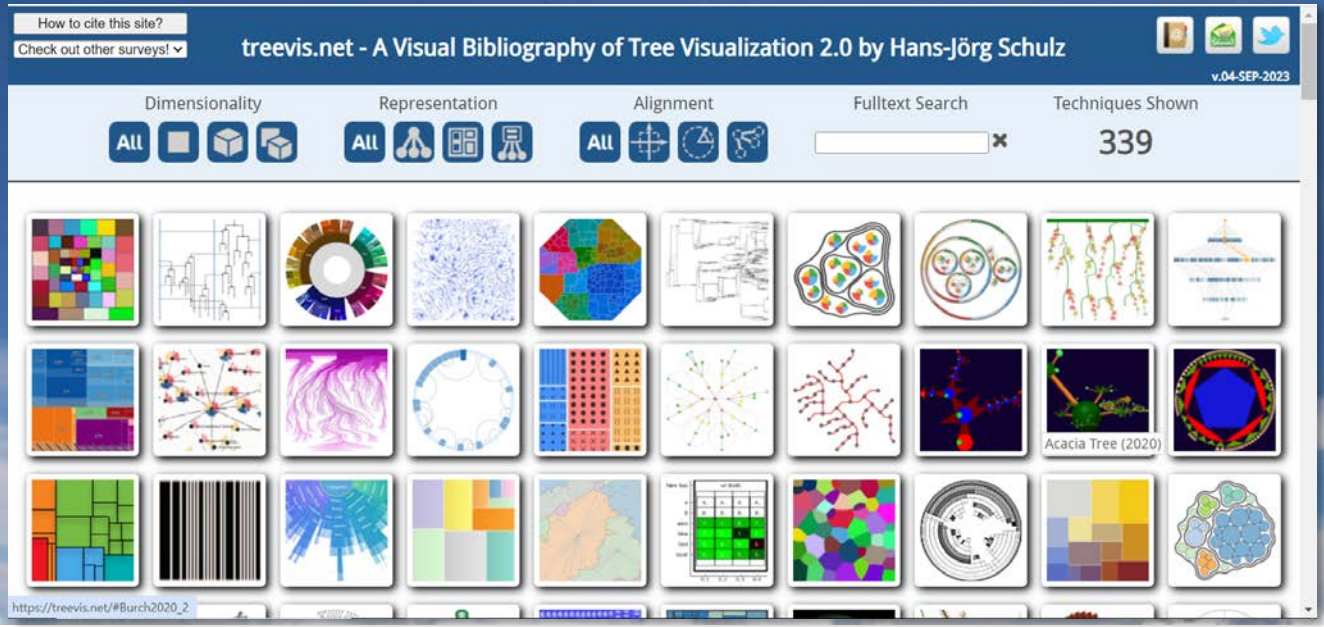
More SPARQL examples

- Panama papers
  - EP
    - http://data.ontotext.com/sparql
  - in RDF
    - ftp://ftp.ontotext.com/pub/leaks/rdf/rdf.zip



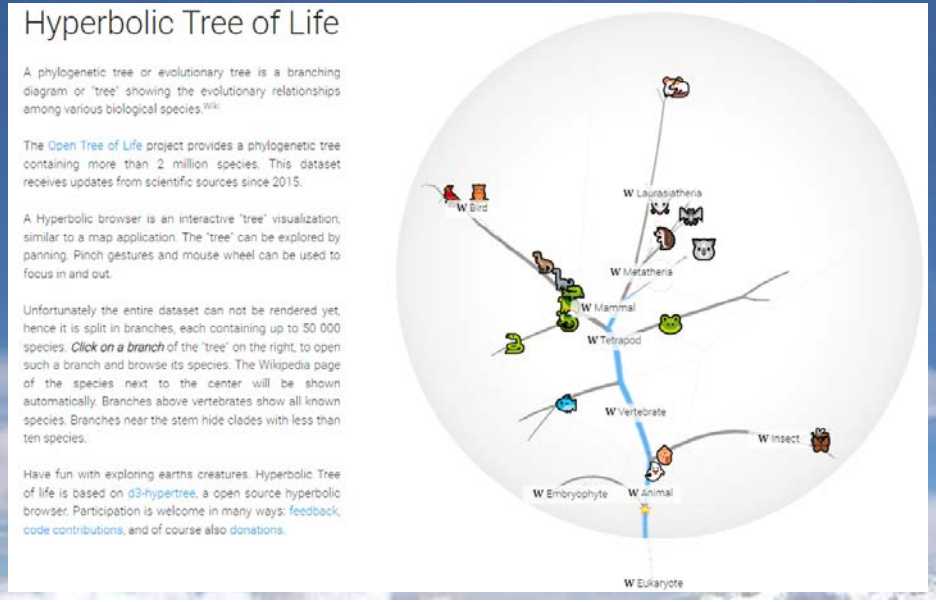
More visualizations

- Visualization of trees
- Treevis.net. 339 visualization methods as of April 2024



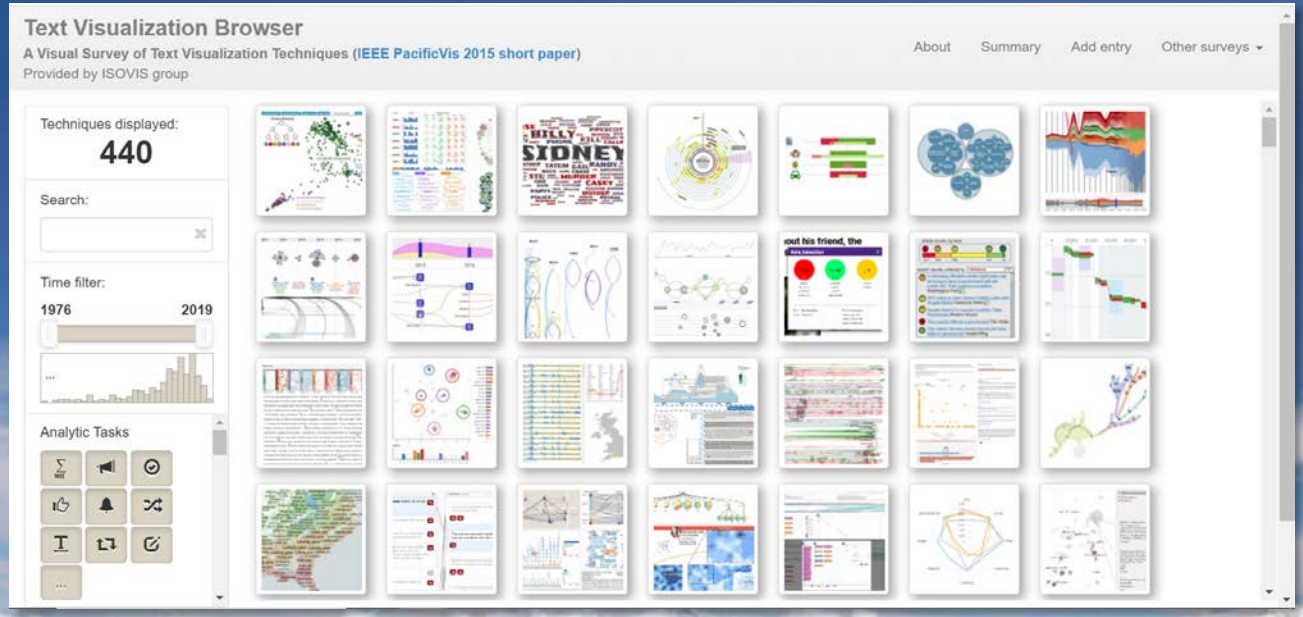
More visualizations

- Visualization of trees
- Hyperbolic trees
  - Michael Glatzhofer (https://glouwa.github.io/d3-hypertree-examples/hyperbolictree-slides)

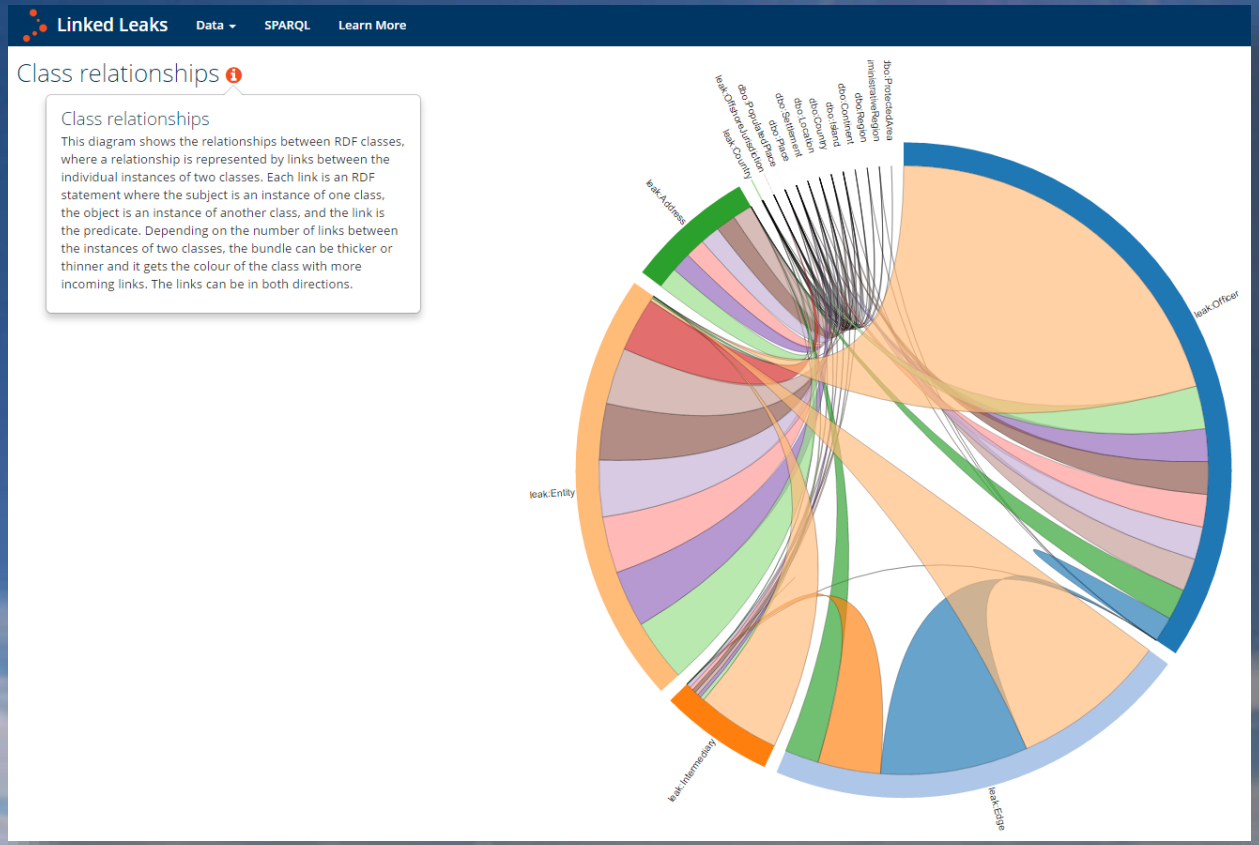


More visualizations

- Visualization of texts
- textvis. 440 visualization methods as of April 2024

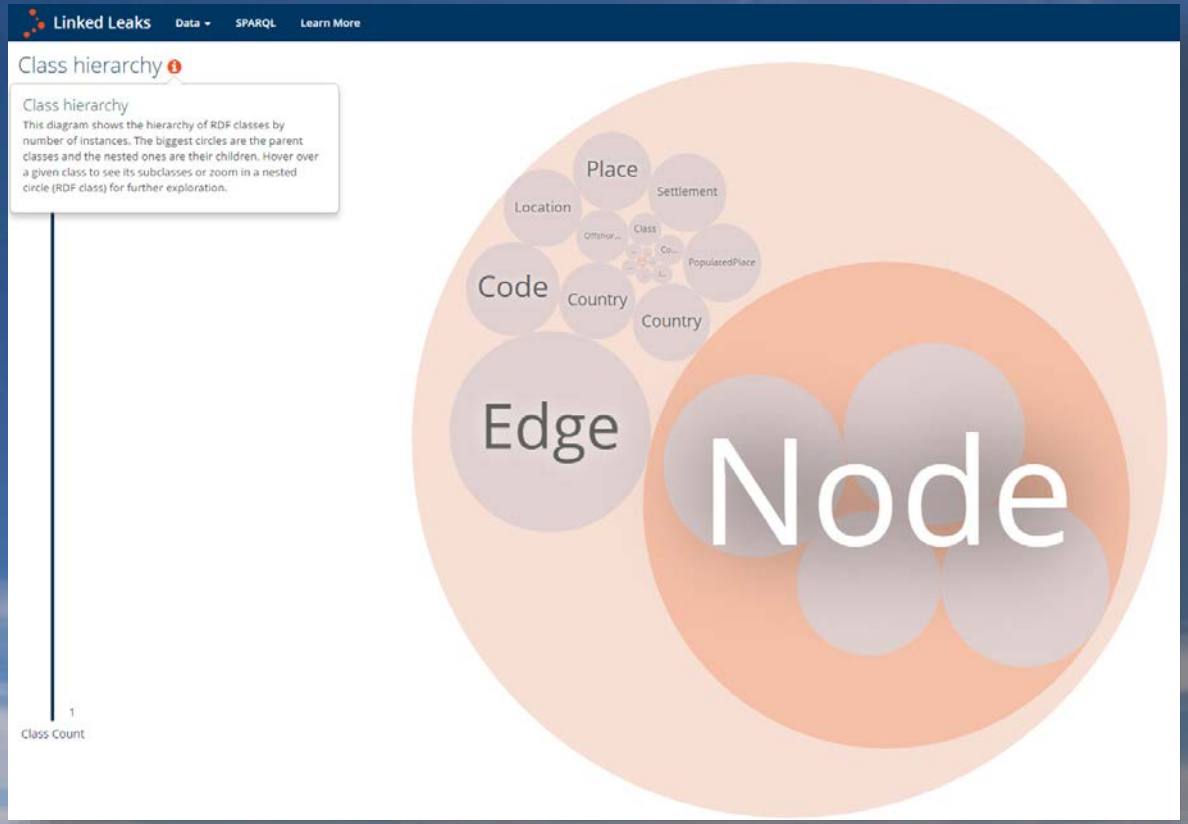


More visualizations



See http://data.ontotext.com/sparql

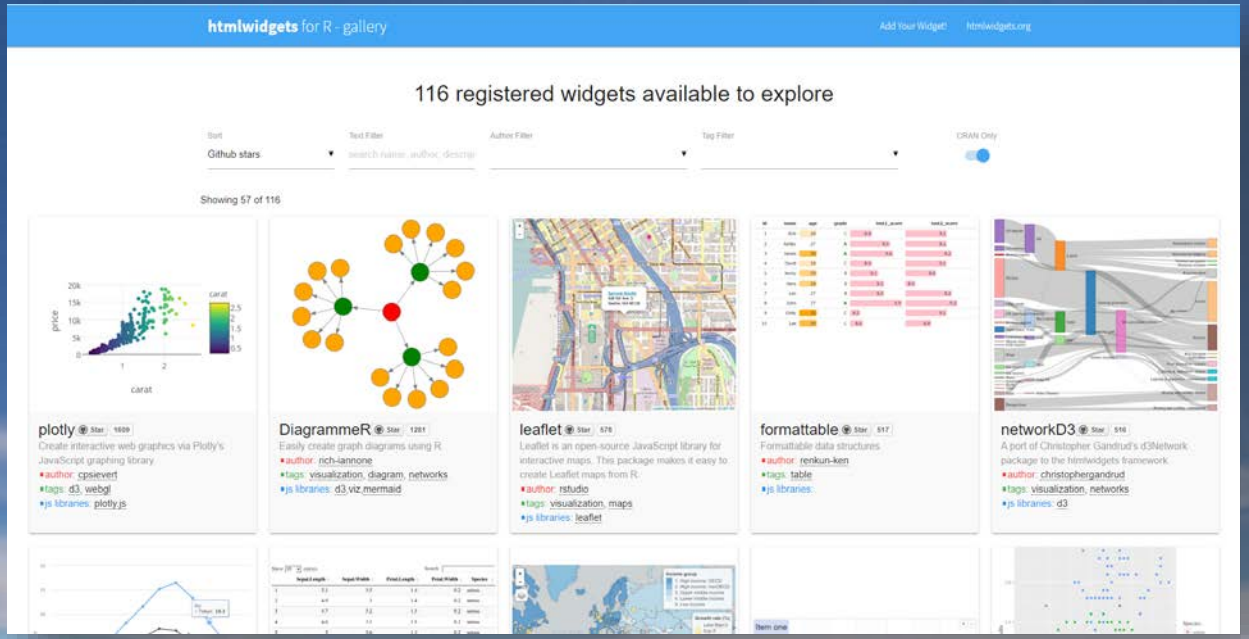
More visualizations



See http://data.ontotext.com/sparql

More visualizations

- The coolest for R: package htmlwidgets

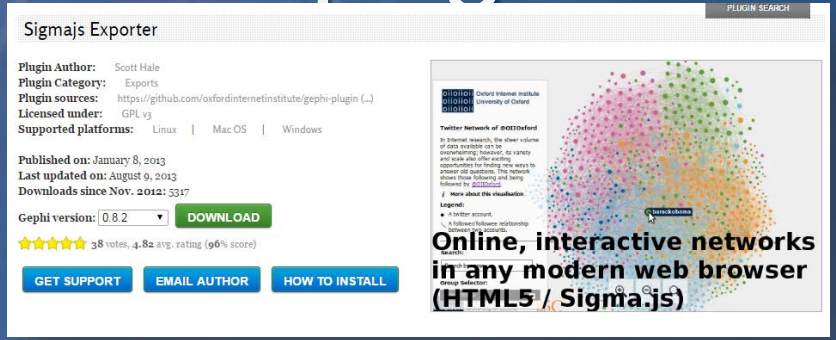


Displaying on a web page

- Sigma sigmajs.org
  - Javascript
  - Reads .gexf files ☺
  - R package
    - github.com/jjallaire/sigma

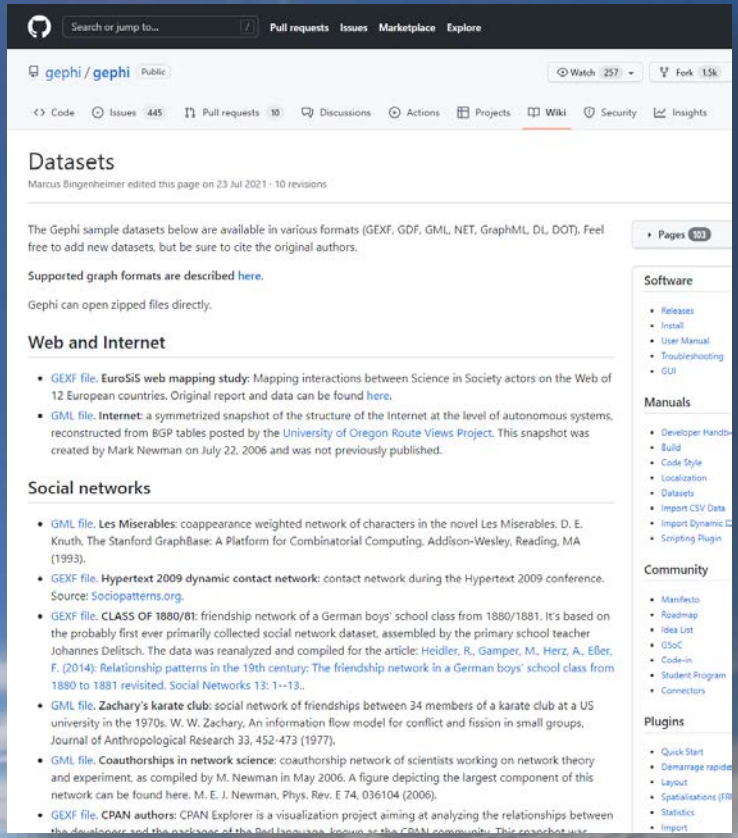
Displaying on a web page

- Gephi exporter for Sigma
  - Generates all the html/js/json code in a given folder
    - Requires a html server (like Tomcat)
      - Does not work as a html file on the browser
  - Limitation: No labels ☹
  - Installation
    - Gephi 0.8.2 vía Marketplace
    - Gephi 0.9.1 vía Tools→Plugins→ Available plugins



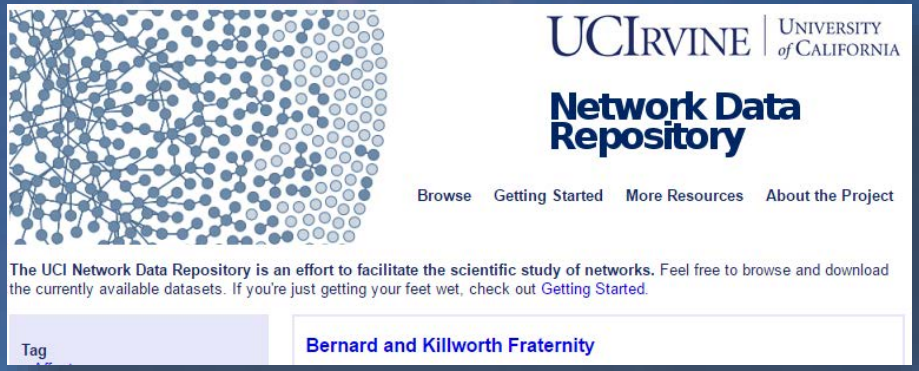
Gephi datasets

- Gephi Wiki “Datasets” page (in Github)
  - https://github.com/gephi/gephi/wiki/Datasets
- Categories:
  - Web and Internet
  - Social networks
  - Biological networks
  - Infrastructure networks
  - Other networks



More datasets

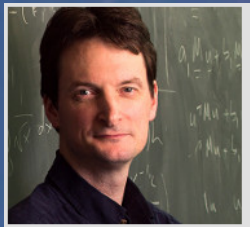
- UCI Network data repository
  - https://networkdata.ics.uci.edu/



- Nexus (igraph.org)
  - http://nexus.igraph.org/api/dataset_info

Even more datasets

- Mark Newman datasets
  - http://www-personal.umich.edu/~mejn/netdata/



Mark Newman

Anatol Rapoport Distinguished University Professor of Physics
Department of Physics and Center for the Study of Complex Systems
University of Michigan

External Faculty
Santa Fe Institute

Network data

This page contains links to some network data sets I've compiled over the years. All of these are free already made the data freely available, or that I have consulted the authors and received permission please cite the original sources.

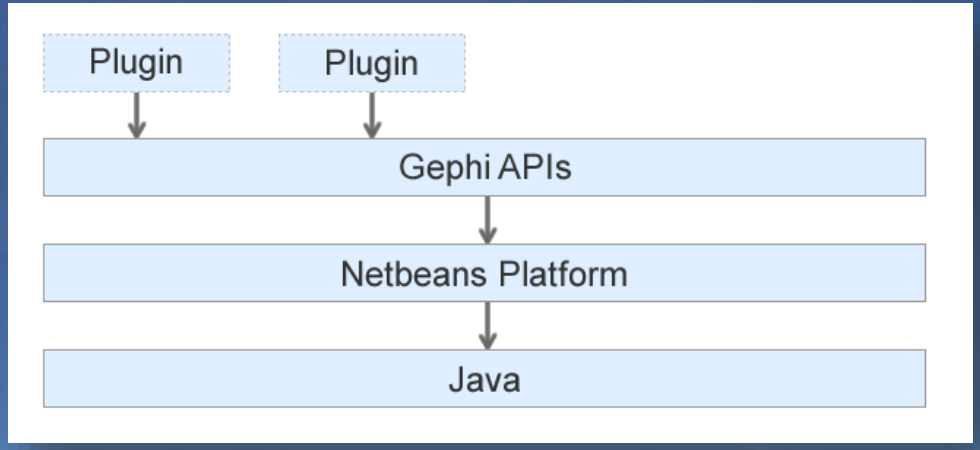
The data sets are in GML format. For a description of GML see here. GML can be read by many net that will read the files into a data structure. It's available here. There are many features of GML not s Python parser for GML available as part of the NetworkX package here and another in the igraph pa software (Java, C++, Perl, R, Matlab, etc.) that reads GML, let me know.

Data sets

- Zachary's karate club: social network of friendships between 34 members of a karate club at a conflict and fission in small groups, *Journal of Anthropological Research* **33**, 452-473 (1977).
- Les Miserables: coappearance network of characters in the novel *Les Miserables*. Please cite I Addison-Wesley, Reading, MA (1993).
- Word adjacencies: adjacency network of common adjectives and nouns in the novel *David Co* (2006).
- American College football: network of American football games between Division IA college *Natl. Acad. Sci. USA* **99**, 7821-7826 (2002).
- Dolphin social network: an undirected social network of frequent associations between 62 dol K. Schneider, O. J. Boisseau, P. Haase, E. Slooten, and S. M. Dawson, *Behavioral Ecology an* these data on this web site.

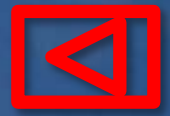
Gephi API

- Built on top of NetBeans platform
  - Thread-safe
  - Multi-view, host sub graphs



- More at Gephi.org (https://github.com/gephi/gephi/wiki/How-to-build-Gephi)

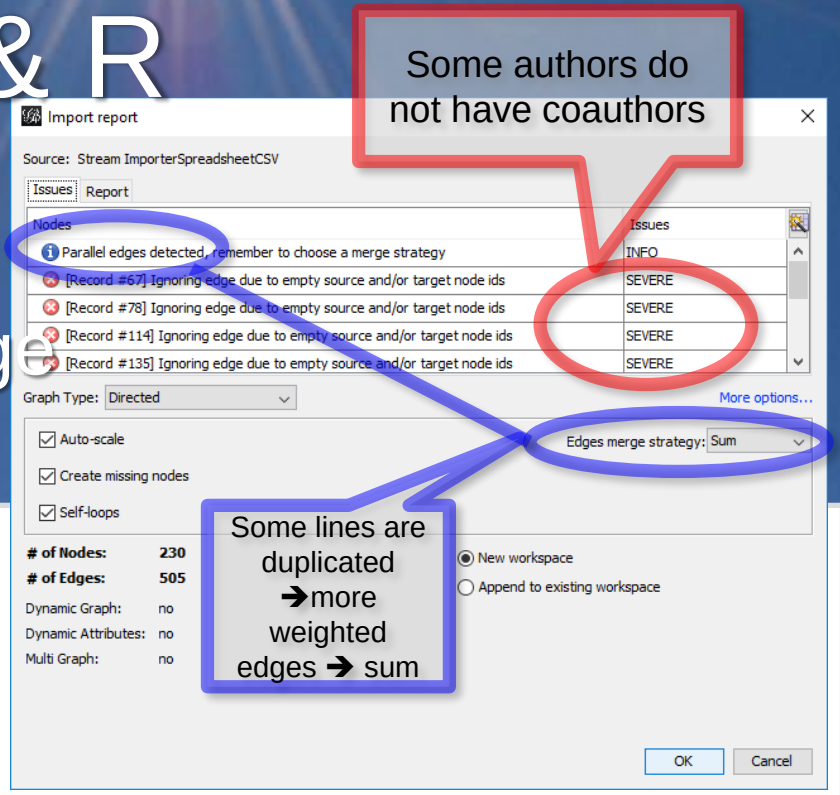
Gephi & R



- From R:
  - Create a graph with the igraph package
    - This package creates igraph objects
  - Write the igraph object to an graphML file
  - Private (non CRAN) package ggraph (grammar's graph, like ggplot2 vs plot).
    - Benefits: igraph+bigrams+ggforce+dendrogram+Hierarchical Edge Bundles + Treemaps + animations…
    - Can convert an igraph to a ggraph
- From Gephi:
  - Read the graphML file
- An example here

Gephi & R

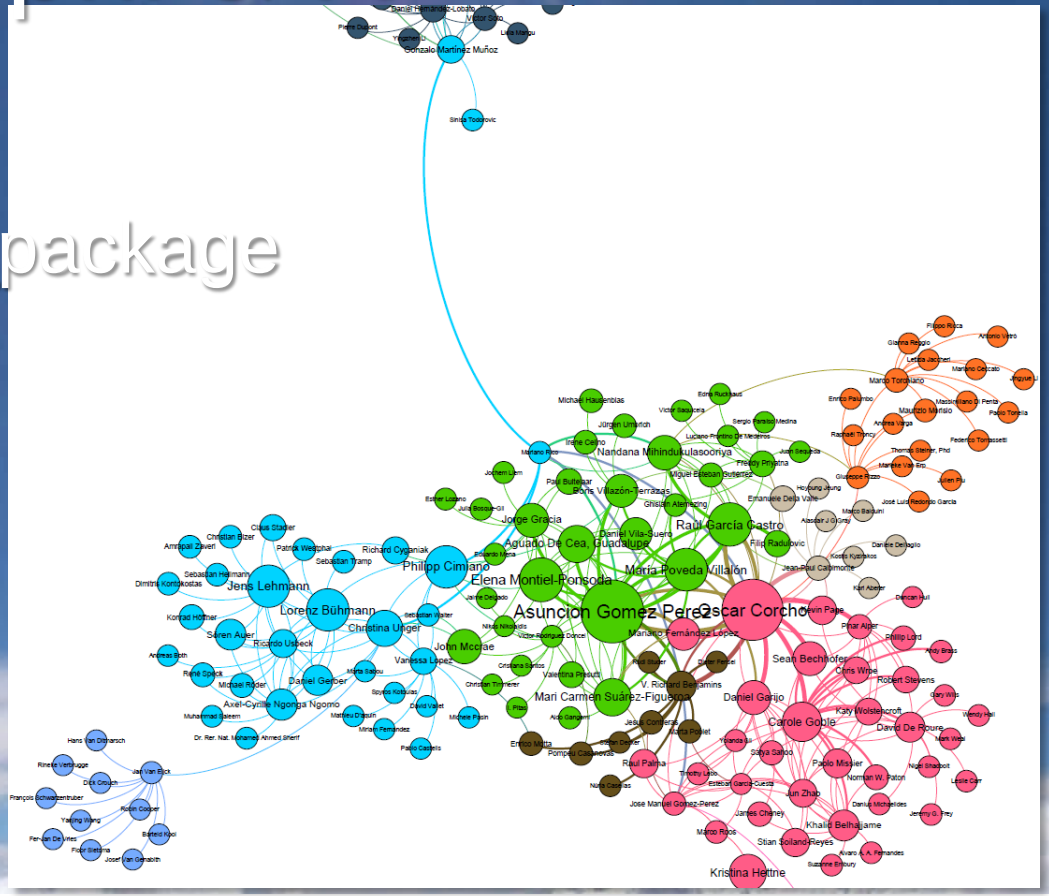
- From R:
  - With the scholar package
    - Export the edges list




```
library(scholar)

#Network of coauthorship of Mariano Rico
co <- get_coauthors("hl5WPQEAAAAJ&hl", 10, 2)

write.table(co, file="coauthorsnet.ok.csv",
            quote = F, row.names = F,
            col.names = c("Source", "Target"), #To be readable by Gephi as 'edges table'
            fileEncoding="UTF-8", sep = ";"    #Authors can have commas
           )
```

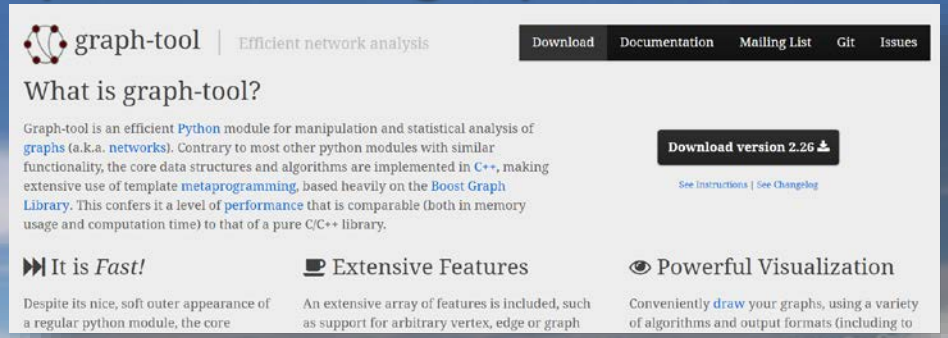
Gephi & R

- From R:
  - With the scholar package
    - Result



For Python fans

- Igraph
  - Also available for Python fans
- graph-tool
  - ToDo: make a comparison to Igraph



Graphs from R

IGRAPH

igraph

- Table of Contents
  - Basics
    - Graph structure
    - Measures
      - Diameter
      - Degree
      - Edge connectivity
      - Edge betweenness
  - Selecting
  - Automatic graphs
  - Layouts para grafos grandes
    - DRL (Distributed Recursive Layout)
    - LGL (Large Graph Layout)
    - Reingold-Tilford tree layout (useful for (almost) tree-like graphs)

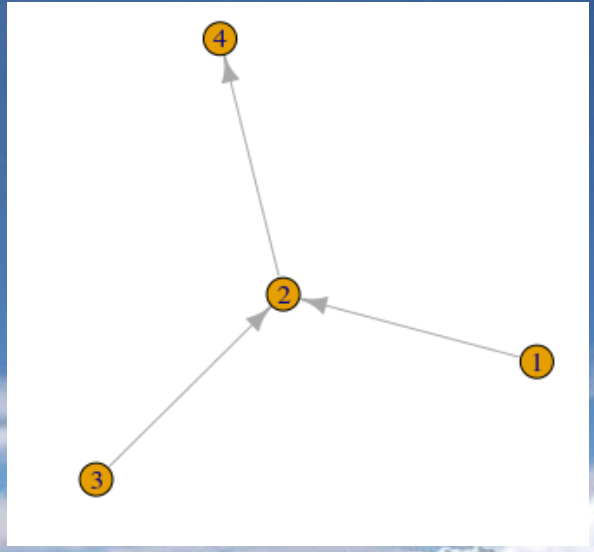
igraph

- Well known software
  - Versions for Python, C and R.
- No good manual ☹
  - But it has a Spanish online tutorial for R

Igraph basics

- A graph is
  - Set of nodes (vertices, singular vertex)
  - Set of arcs (edges).

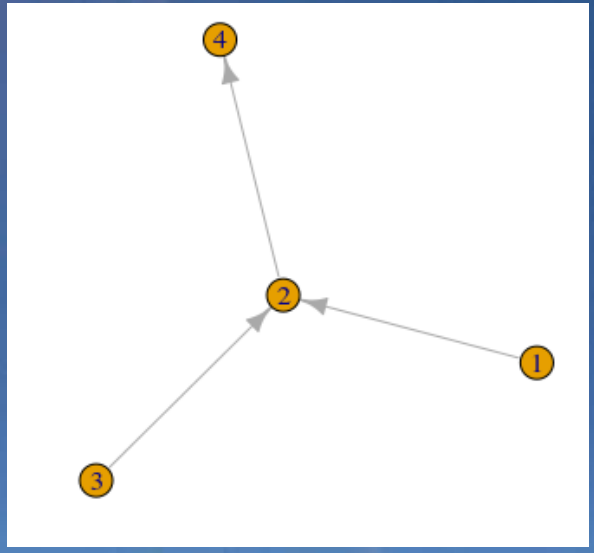
```
library(igraph)

#Graphs with the given list of edges
edges <- c(1,2, 3,2, 2,4)
g<-graph(edges, n=max(edges), directed=TRUE)
plot(g)
```



Igraph basics

- Graph structure

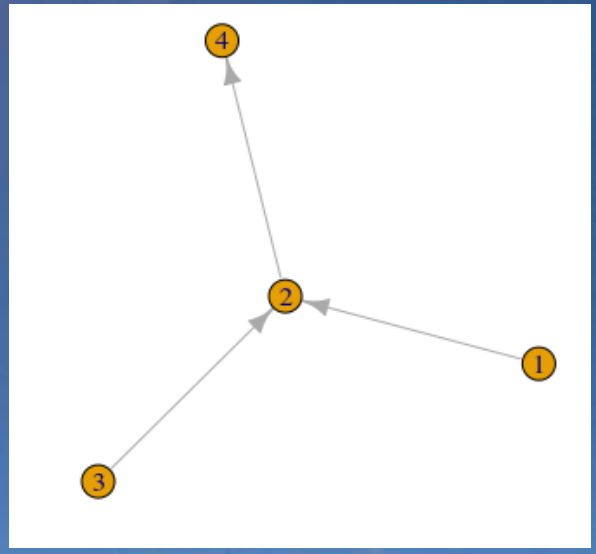
```
library(igraph)

#Graphs with the given list of edges
edges <- c(1,2, 3,2, 2,4)
g<-graph(edges, n=max(edges), directed=TRUE)

#Number of vertices (nodes)
vcount(g) #Returns 4

#Number of edges
ecount(g) #Returns 3

#The vertices
V(g) #Returns an object igraph.vs (“vertex sequence”)
> [1] 1 2 3 4
#The edges
E(g) #Returns an object igraph.es (“edge sequence”)
>[1] 1->2 3->2 2->4
```



Igraph basics

- Graph structure

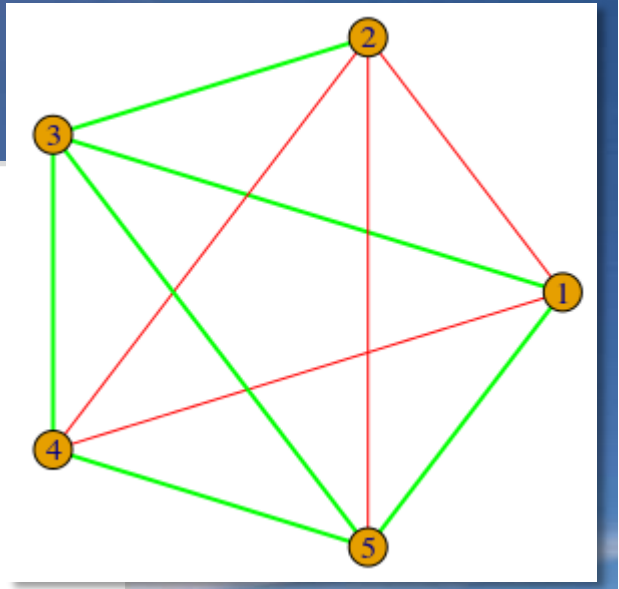
```
library(igraph)

#Graphs with the given list of edges
edges <- c(1,2, 3,2, 2,4)
g<-graph(edges, n=max(edges), directed=TRUE)

#Neighbors
neighbors(g,
          V(g)[1],      # 1 2 3 4
          mode = "all") # "in" and "out"
>[1] 2

neighbors(g, V(g)[2], mode = "all") #All nodes from/to node 2
>[1] 1 3 4

#Connection
are.connected(g, V(g)[1], V(g)[3])
> [1] FALSE
```



Igraph basics

- Graph attributes
  - For vertices and edges

```
g <- graph.full(5) #Fully connected

set.seed(666) #To get always the same graph
E(g)$weight <- runif(ecount(g)) #rand uniform between 0 and 1
E(g)$width <- 1
E(g)$color <- "red"
E(g)[ weight < 0.5 ]$width <- 2
E(g)[ weight < 0.5 ]$color <- "green"

plot(g,
     layout=layout.circle,
     edge.width=E(g)$width,
     edge.color= E(g)$color)
```



Igraph basics

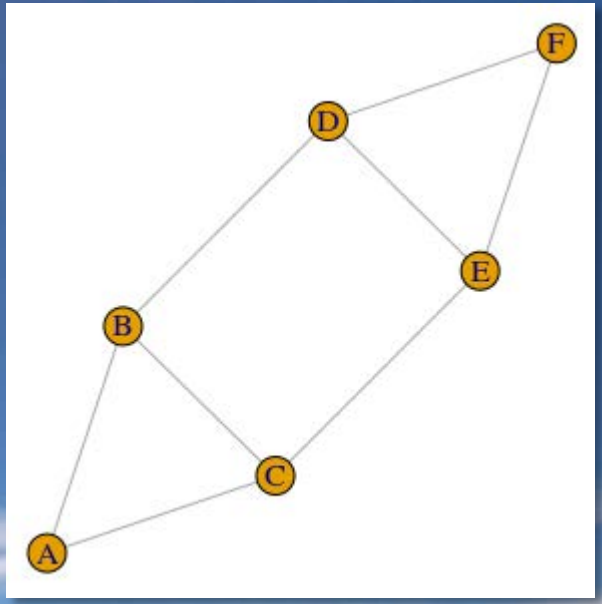
- Plot params

NODES	
vertex.color	Node color
vertex.frame.color	Node border color
vertex.shape	One of “none”, “circle”, “square”, “csquare”, “rectangle” “crectangle”, “vrectangle”, “pie”, “raster”, or “sphere”
vertex.size	Size of the node (default is 15)
vertex.size2	The second size of the node (e.g. for a rectangle)
vertex.label	Character vector used to label the nodes
vertex.label.family	Font family of the label (e.g.“Times”, “Helvetica”)
vertex.label.font	Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol
vertex.label.cex	Font size (multiplication factor, device-dependent)
vertex.label.dist	Distance between the label and the vertex
vertex.label.degree	The position of the label in relation to the vertex, where 0 right, “pi” is left, “pi/2” is below, and “-pi/2” is above
EDGES	
edge.color	Edge color
edge.width	Edge width, defaults to 1
edge.arrow.size	Arrow size, defaults to 1
edge.arrow.width	Arrow width, defaults to 1
edge.lty	Line type, could be 0 or “blank”, 1 or “solid”, 2 or “dashed”, 3 or “dotted”, 4 or “dotdash”, 5 or “longdash”, 6 or “twodash”
edge.label	Character vector used to label edges
edge.label.family	Font family of the label (e.g.“Times”, “Helvetica”)
edge.label.font	Font: 1 plain, 2 bold, 3, italic, 4 bold italic, 5 symbol
edge.label.cex	Font size for edge labels
edge.curved	Edge curvature, range 0-1 (FALSE sets it to 0, TRUE to 0.5)
arrow.mode	Vector specifying whether edges should have arrows, possible values: 0 no arrow, 1 back, 2 forward, 3 both
OTHER	
margin	Empty space margins around the plot, vector with length 4
frame	if TRUE, the plot will be framed
main	If set, adds a title to the plot
sub	If set, adds a subtitle to the plot

Igraph basics

- Measures

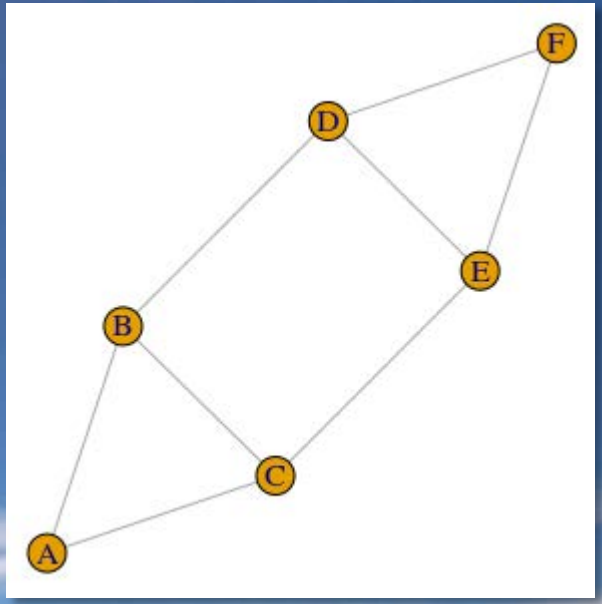
```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)
plot(g)

diameter(g) #length of the longest geodesic
> [1] 3

get_diameter(g) #path with the actual diameter
+ 4/6 vertices, named:
[1] A B D F

farthest_vertices(g) #a list with
$vertices                  # the 2 farthest vertices
+ 2/6 vertices, named:
[1] A F
$distance                  # the distance
[1] 3
```



Igraph basics

- Measures: diameter

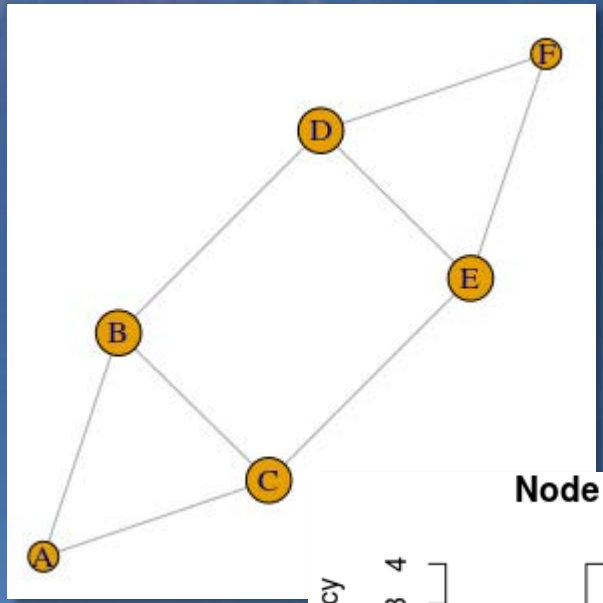
```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)
plot(g)

diameter(g) #length of the longest geodesic
> [1] 3

get_diameter(g) #path with the actual diameter
+ 4/6 vertices, named:
[1] A B D F

farthest_vertices(g) #a list with
$vertices                  # the 2 farthest vertices
+ 2/6 vertices, named:
[1] A F
$distance                  # the distance
[1] 3
```

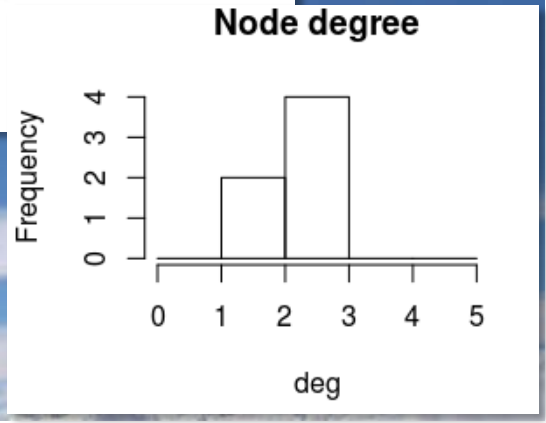


Igraph basics

- Measures: degree

```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)
plot(g)

deg <- degree(g, mode="all")
#deg
#A B D F E C
#2 3 3 2 3 3   # 2 nodes with degree 2
              # 4 nodes with degree 3

plot(g, vertex.size=deg*6)

#The histogram
hist(deg,
     breaks=1:vcount(g)-1,
     main="Node degree")
```

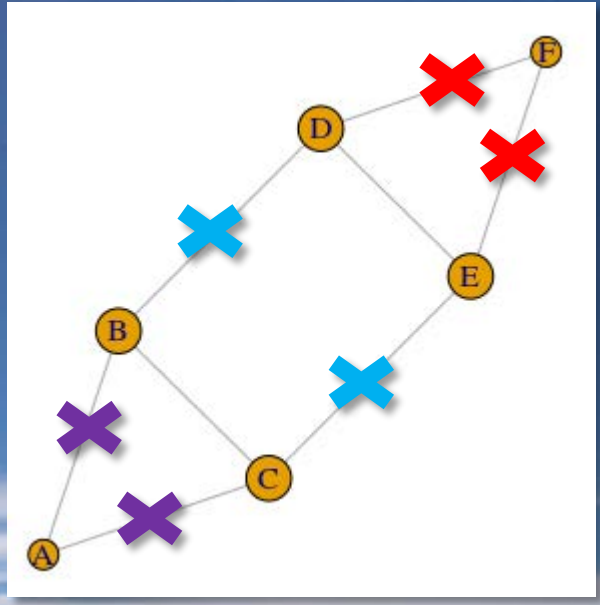




Igraph basics

- Measures: edge connectivity

```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)
plot(g)

#Minimum number of edges needed to remove
#to obtain a graph which is not strongly connected.
edge_connectivity(g)
#[1] 2

#You can get the edges with min_cut.
#WARNING! If several min cuts are…
# …available returns the first one ONLY.
min_cut(g, value.only = F)$cut
#+ 2/8 edges (vertex names):
#[1] D--F F--E
```

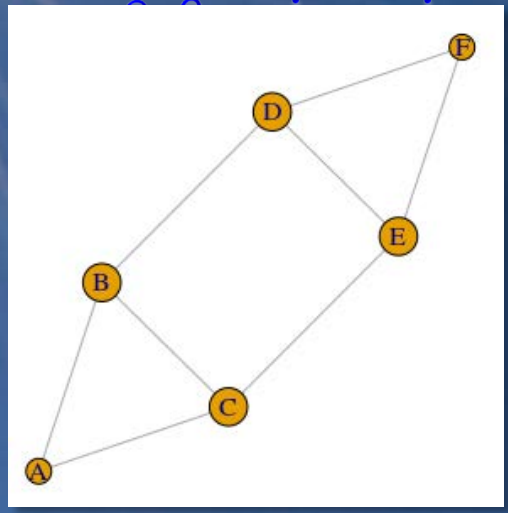


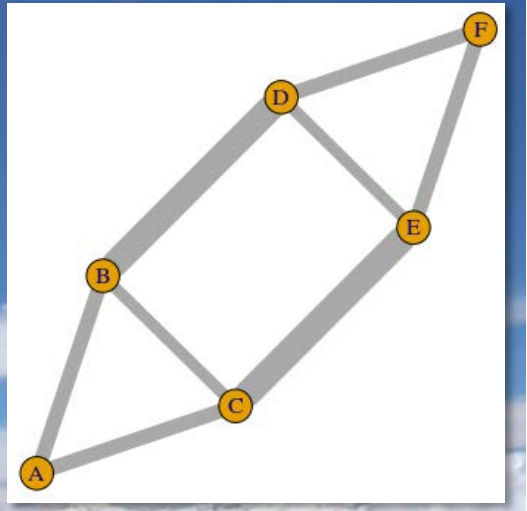
#VizLinkedData

Igraph basics

- Measures: edge betweenness

```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)

#is (roughly) defined by the number of geodesics
#(shortest paths) going through an edge.
#Similar to vertex betweenness.
betw <- edge_betweenness(g,
                         directed = FALSE)
betw
#[1] 2.5  2.5  4.5  2.0  2.5  2.0  2.5  4.5
E(g)
#[1] A--B A--C B--D B--C D--F D--E F--E E–C

plot(g,
     edge.width=betw*4)
```



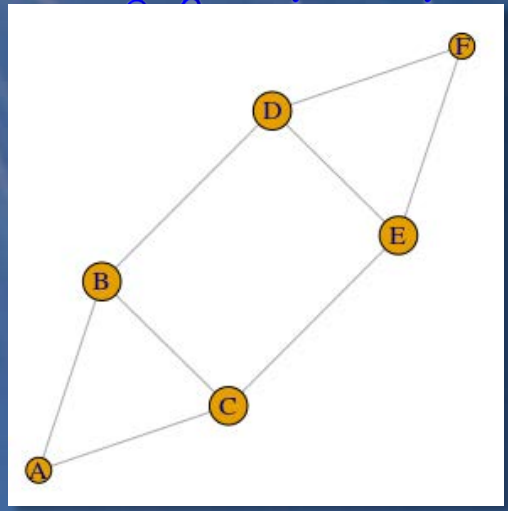


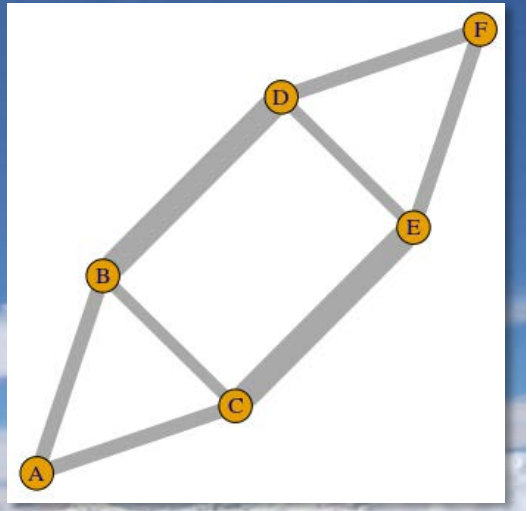
#VizLinkedData

Igraph basics

- Attributes

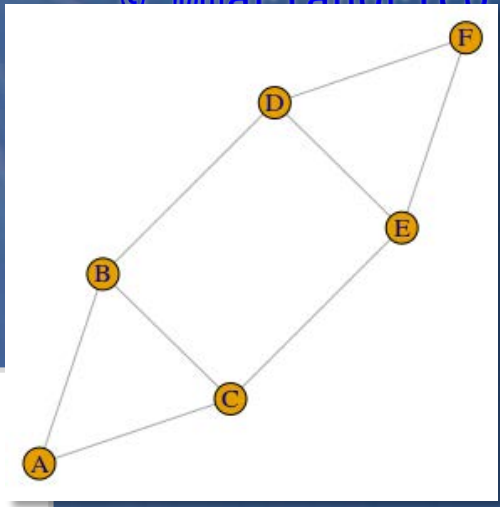
```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)

#is (roughly) defined by the number of geodesics
#(shortest paths) going through an edge.
#Similar to vertex betweenness.
betw <- edge_betweenness(g,
                         directed = FALSE)
betw
#[1] 2.5  2.5  4.5  2.0  2.5  2.0  2.5  4.5
E(g)
#[1] A--B A--C B--D B--C D--F D--E F--E E–C
E(g)$betw <- betw
plot(g)
```





Selecting

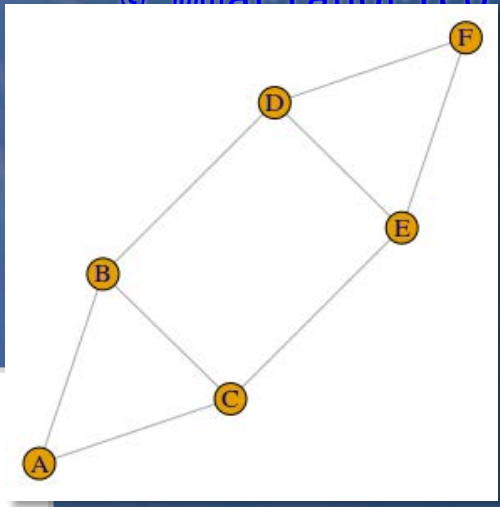
- incident(), from() and to()




```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)

#incident takes a vertex sequence, and selects all edges that have at least
one incident vertex in the vertex sequence.
#mode = c("all", "out", "in", "total")

incident(g, V(g)[name == "C"], mode="all")
     # + 3/8 edges (vertex names):
     # [1] A--C B--C E–C
```

Selecting

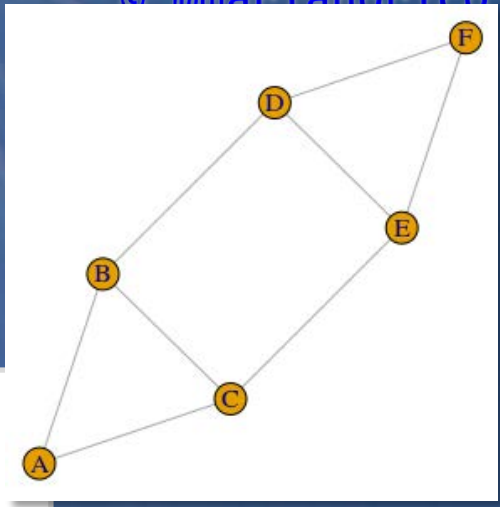
- incident(), from() and to()




```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)

#g[[from=seq]] returns vertices
#g[[from=seq, edges=TRUE]] returns edges
g[[from = 1:2]] #equivalent to g[[from = c("A", "B")]]
     #$A
     #+ 2/6 vertices, named:
     #[1] B C
     #$B
     #+ 3/6 vertices, named:
     #[1] A D C
g[[from = 1:2, edges = T]] #returns the edges from nodes A and B
```

Selecting

- incident(), from() and to()

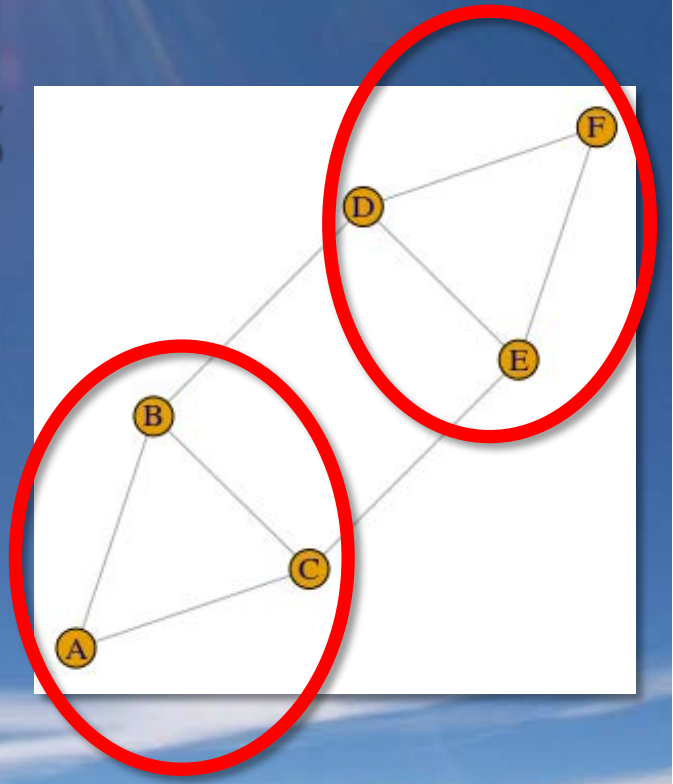



```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)

#g[[to=seq]] returns vertices
#g[[to=seq, edges=TRUE]] returns edges
g[[to = c("E")]] # vertices to node E
     #$E
     #+ 3/6 vertices, named:
     #[1] D F C

g[[to = c("E"), edges=T]] #edges to node E
$E
+ 3/8 edges (vertex names):
[1] D--E F--E E--C
```

Selecting

- Operators %--%, %->%, %<-%

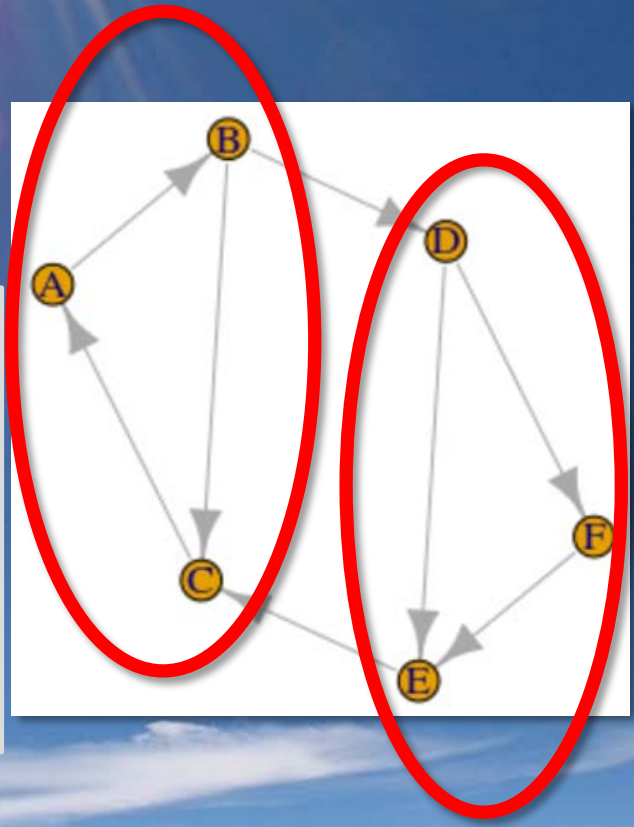
```
set.seed(222)
g <- graph_from_literal(A-B-D-F-E-C-A, B-C, D-E)

# Select all edges between two sets of vertices.
# vertices between set (A-B-C) and set(D-E-F)
E(g)[V(g)[name <= "C"] %--% V(g)[name >= "D"]]
     #+ 2/8 edges (vertex names):
     #[1] B--D E—C
```



Selecting

- Operators %--%, %->%, %<-%

```
set.seed(666)
g <- graph_from_literal(A-+B-+D-+F-+E-+C-+A, B-+C, D-+E)

# Edges from set (A-B-C) to set(D-E-F).
E(g)[V(g)[name <= "C"] %->% V(g)[name >= "D"]]
     # + 1/8 edge (vertex names):
     # [1] B->D

E(g)[V(g)[name <= "C"] %<-% V(g)[name >= "D"]]
     # + 1/8 edge (vertex names):
     # [1] E->C
```

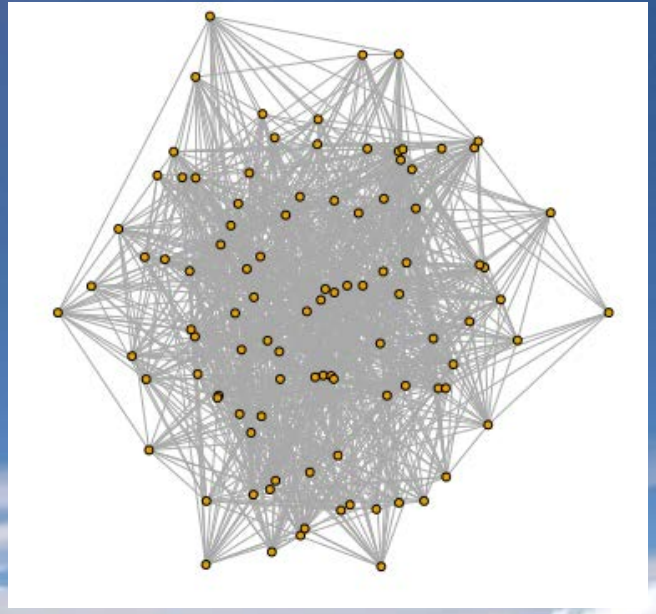


Igraph automatic graphs

- Erdos-Renyi model
  - `sample_gnm()`. G(n, m) model.
    - n vertices, m edges (selected randomly from all possible edges)

```
library(igraph)
set.seed(123)
g <- as.undirected(sample_gnm(n=100, m=1000))
l <- layout_with_drl(g, dim=3) #3D positions
plot(g, layout=l, vertex.size=3, vertex.label=NA)
```

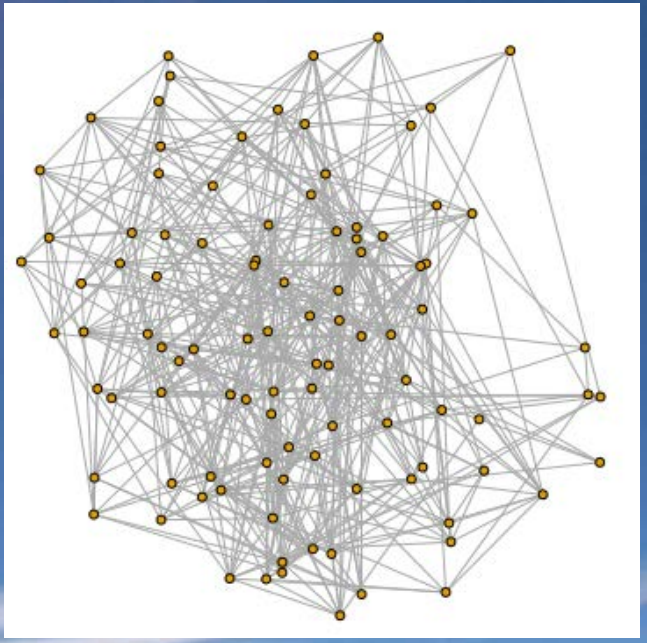
edge_density(g) = 0.202



Igraph automatic graphs

- Erdos-Renyi model
  - `sample_gnp()`. G(n, p) model.
    - n vertices, p probability of creating an edge between 2 random vertices

```
library(igraph)
set.seed(123)
g <- as.undirected(sample_gnp(n=100, p=0.1))
l <- layout_with_drl(g, dim=3) #3D positions
plot(g, layout=l, vertex.size=3, vertex.label=NA)
```



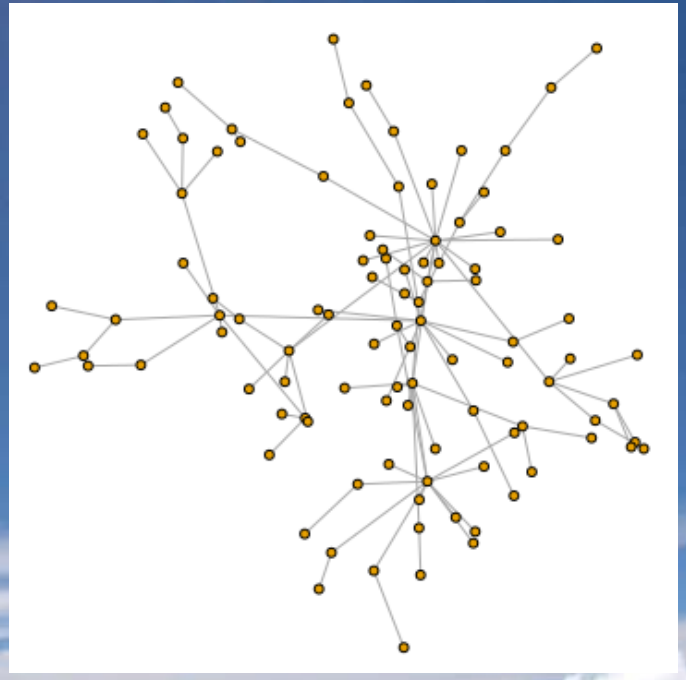
edge_density(g) = 0.096 (~0.1)

Igraph automatic graphs

- Barabasi-Albert model
  - `sample_pa()`.
    - n vertices, many more params (with default values)

```
library(igraph)
set.seed(123)
g <- as.undirected(sample_pa(n=100))
l <- layout_with_drl(g, dim=3) #3D positions
plot(g, layout=l, vertex.size=3, vertex.label=NA)
```

edge_density(g) = 0.02

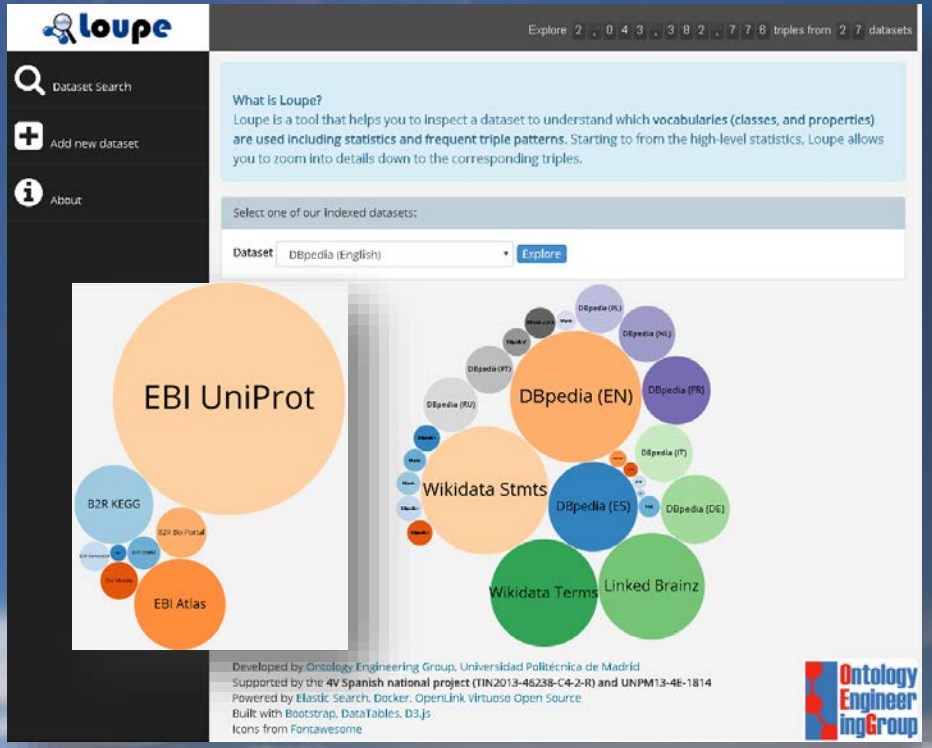


Exploring a given dataset

LOUPE

Loupe

- Explore datasets
  - Two flavors
    - Regular Loupe
      - Popular LOD datasets
        - » 27 datasets
        - » 2.0 billion triples
    - BioLoupe
      - Popular Bio datasets
        - » 8 datasets
        - » 4.8 billion triples



Loupe

- Better an online demo ☺

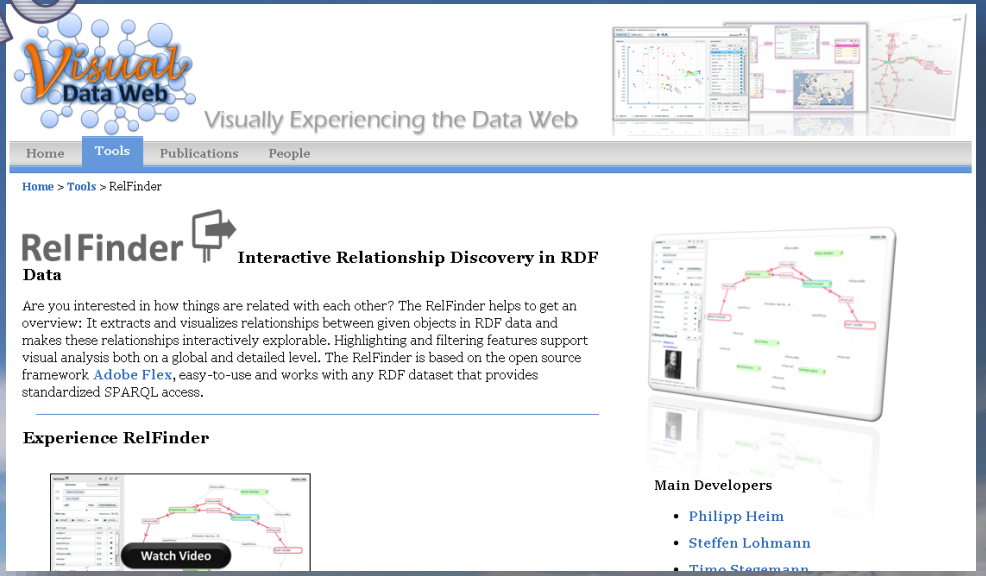
A first glimpse of graphs with

RELFINDER

RelFinder

- A first graphical exploration
- Configurable for any SPARQL endpoint
  - By default: DBpedia
- Go!

Not available now ☹

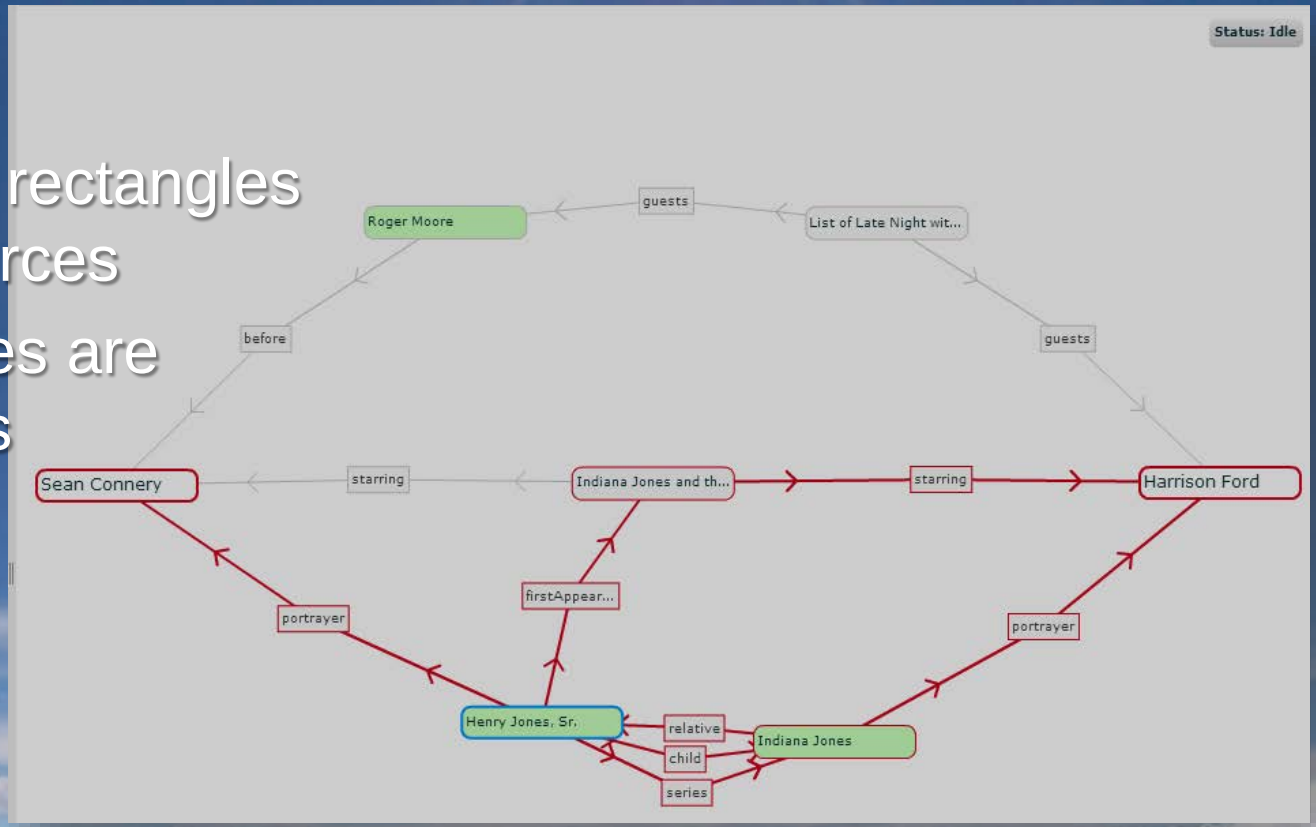


RelFinder

- Look for the relationship between Sean Connery and Harrison Ford

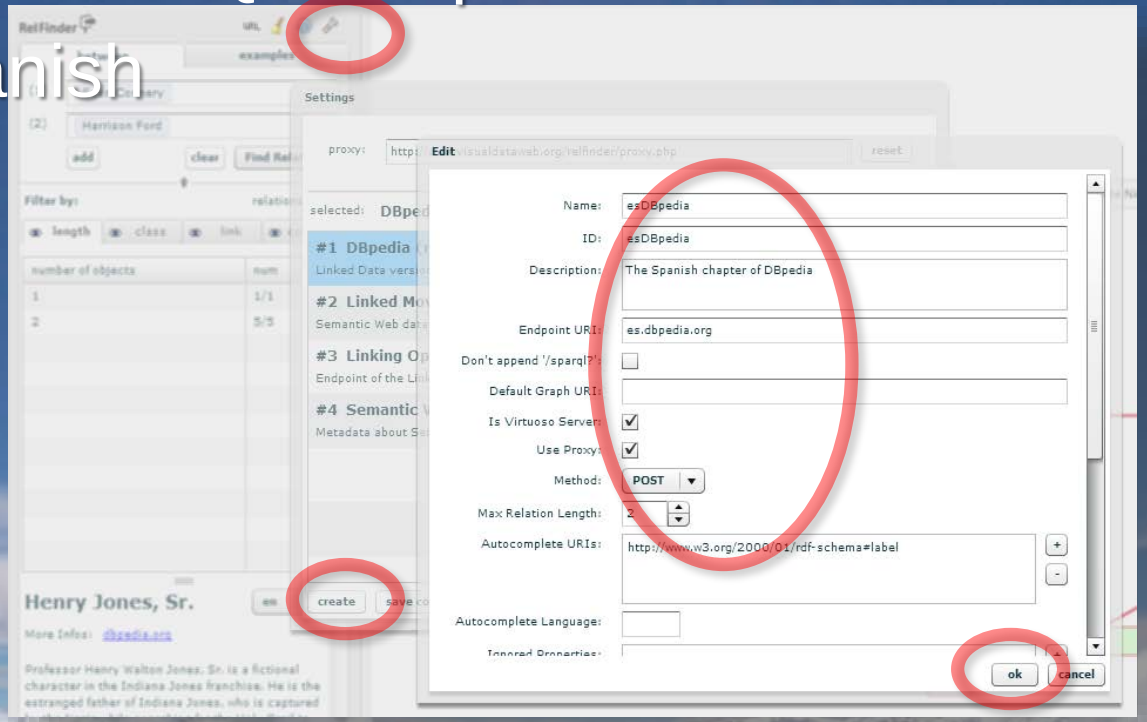
RelFinder

- Result
  - Rounded rectangles are resources
  - Rectangles are properties



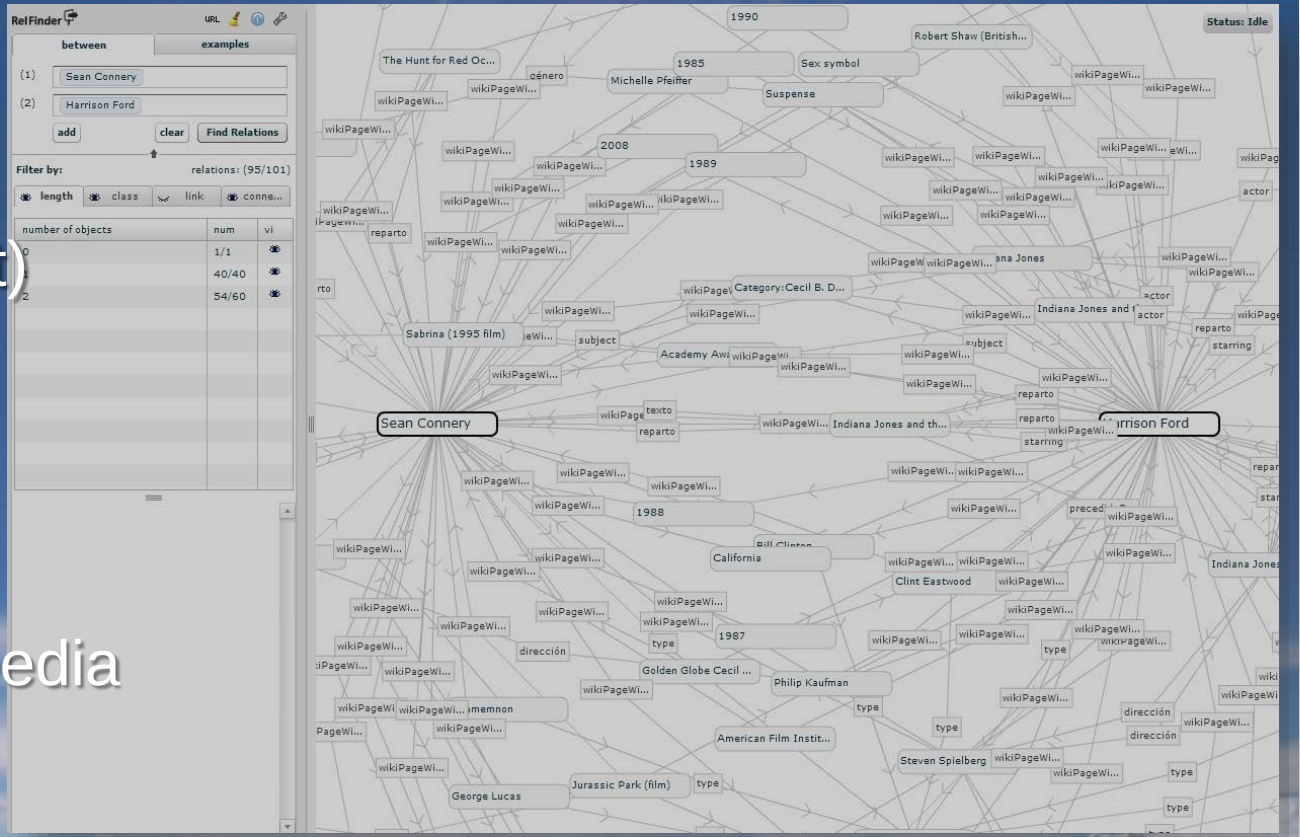
RelFinder

- Setup for other SPARQL endpoints
  - E.g. The Spanish DBpedia



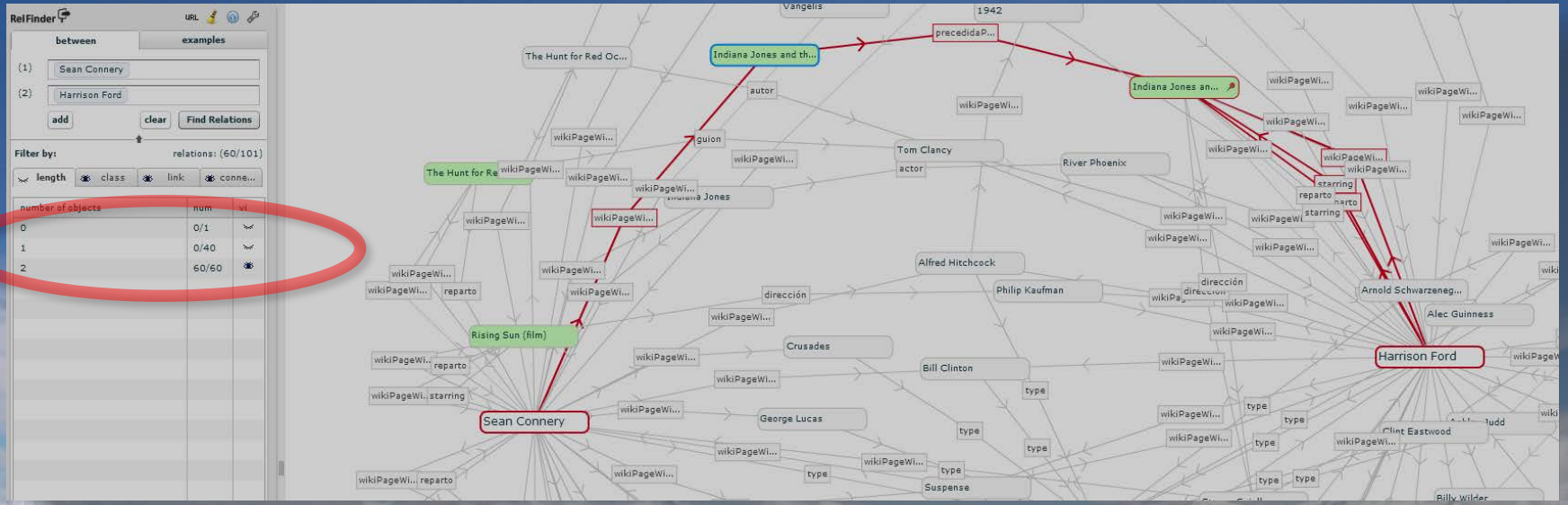
RelFinder

- Result
  - Much more than (default) DBpedia
  - Hypothesis: They use a old private copy of DBpedia



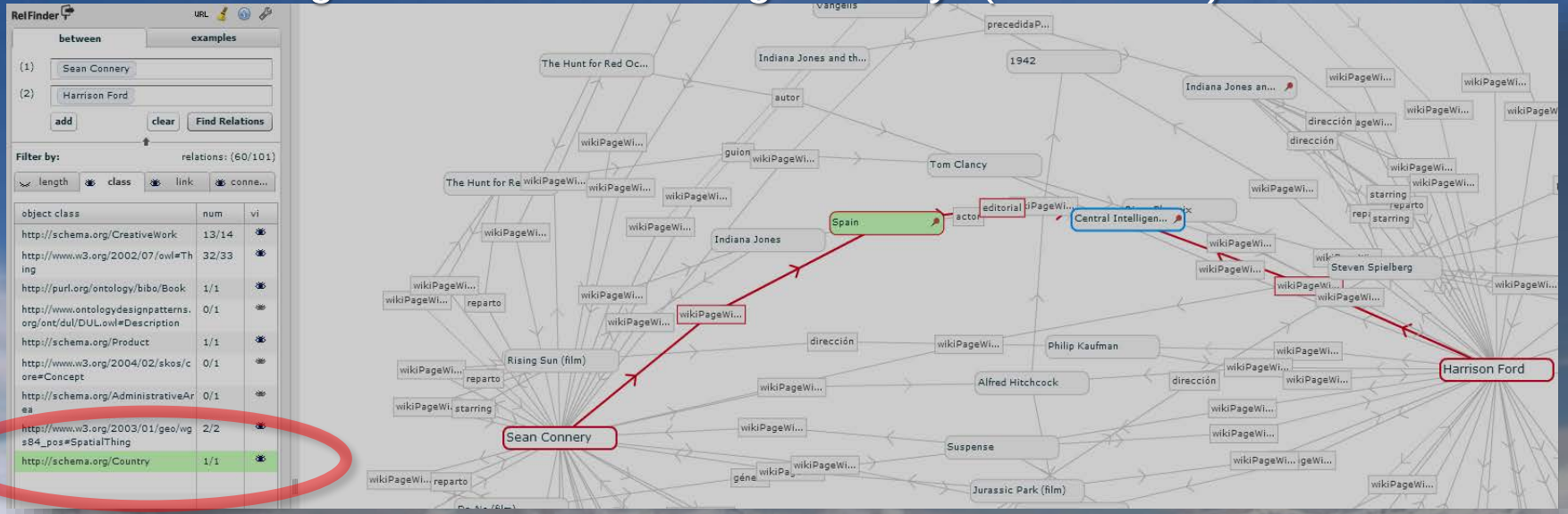
RelFinder

- Filter by
  - Length (number of resources in between)
    - E.g. length = 2 ➔ 2 resources in between



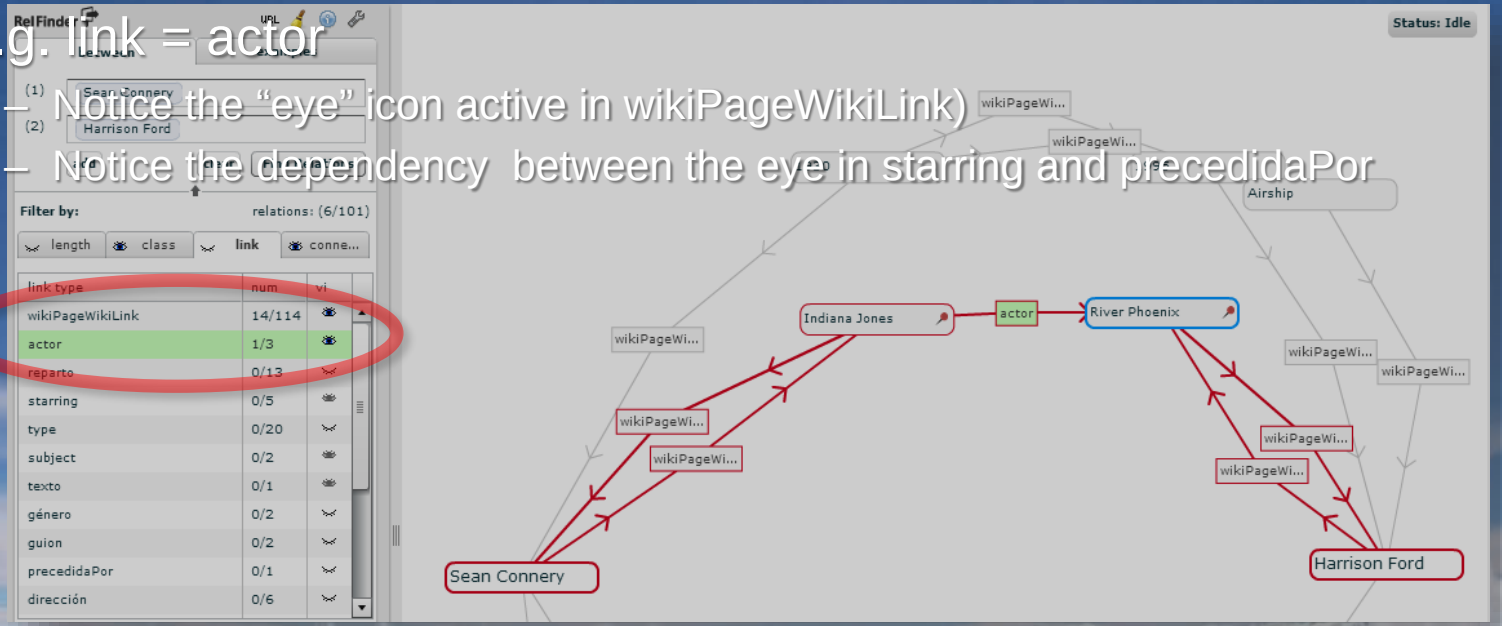
RelFinder

- Filter by
  - Class (list of resource's class)
    - E.g. class = schema.org/Country (num = 1/1)



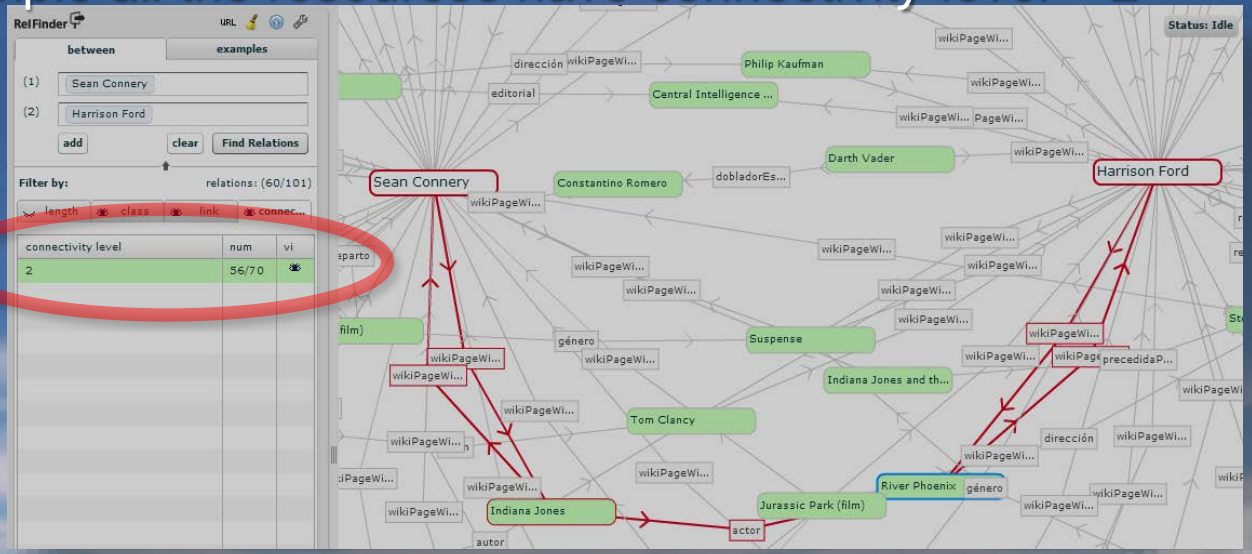
RelFinder

- Filter by
  - Link (list of resource's properties)
    - E.g. link = actor
      - Notice the "eye" icon active in wikiPageWikiLink)
      - Notice the dependency between the eye in starring and precedidaPor



RelFinder

- Filter by
  - Connectivity level (number of resources in between)
    - In this example all the resources have connectivity level = 2



RelFinder

- Limitations
  - Oriented to resources
    - 2 resources (or more)
  - Very noisy
    - properties as graph nodes

After RelFinder

- EveryThingIsConnected
  - Up to 15 hops
    - 2 resources (or more)
  - Creates a movie with the explanation
  - Has a REST API
    - E.g.: Rajoy ←→ Arabidopsis

My record: 11 hops

AS YOU CAN SEE, MARIANO RAJOY
IS CONNECTED TO EVERYTHING IN THIS WORLD,
INCLUDING ARABIDOPSIS THALIANA!

Ordered small graphs

GRUFF

Gruff

- Private product (evaluation license)
- Last version includes access to EPs
  - Local installation or in cloud
- Windows/Linux/Mac

Gruff

- Windows/Linux/Mac

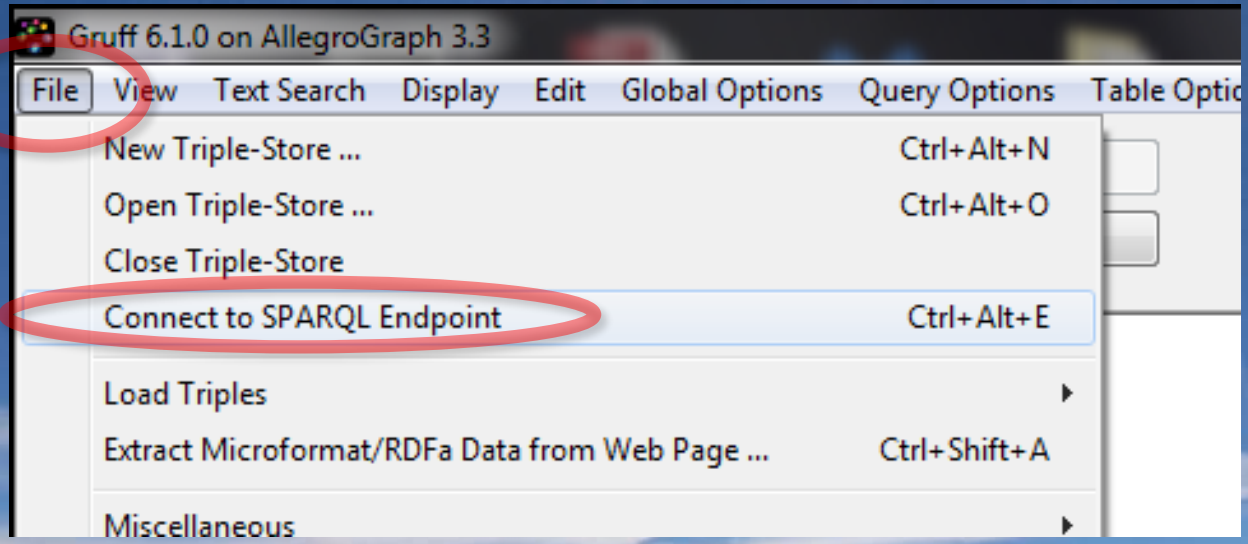
Gruff for AllegroGraph 3.3

A Standalone Gruff with built-in AllegroGraph 3.3, **no Additional Download Required**:

Platform	Download	System Requirements
64-bit Linux	Gruff v6.1.0-AG3.3	glibc 2.4 or newer -- Installation instructions
32-bit Linux	Gruff v6.1.0-AG3.3	glibc 2.3 or newer -- Installation instructions
64-bit Mac OS X	Gruff v6.1.0-AG3.3	Mac OS X 10.6 or newer -- Installation instructions
64-bit Windows	Gruff v6.1.0-AG3.3	Windows Vista or newer -- Installation instructions
32-bit Windows	Gruff v6.1.0-AG3.3	Windows Vista or newer -- Installation instructions

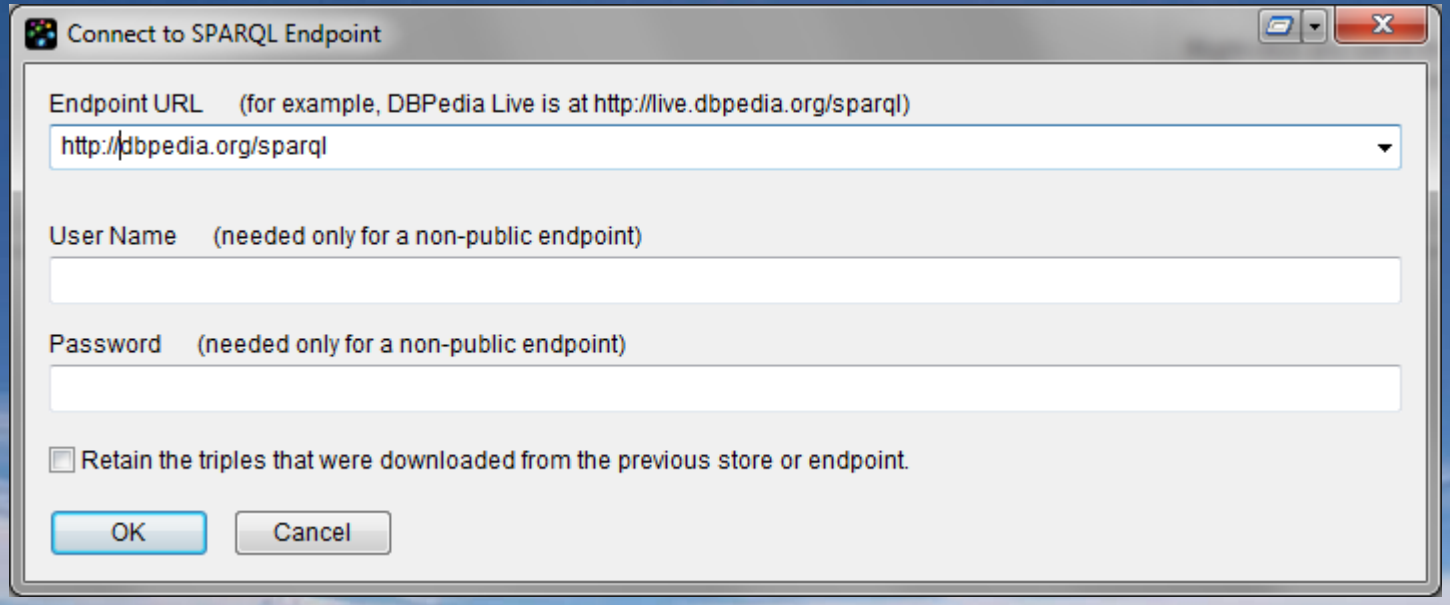
Gruff

- Specify the EP



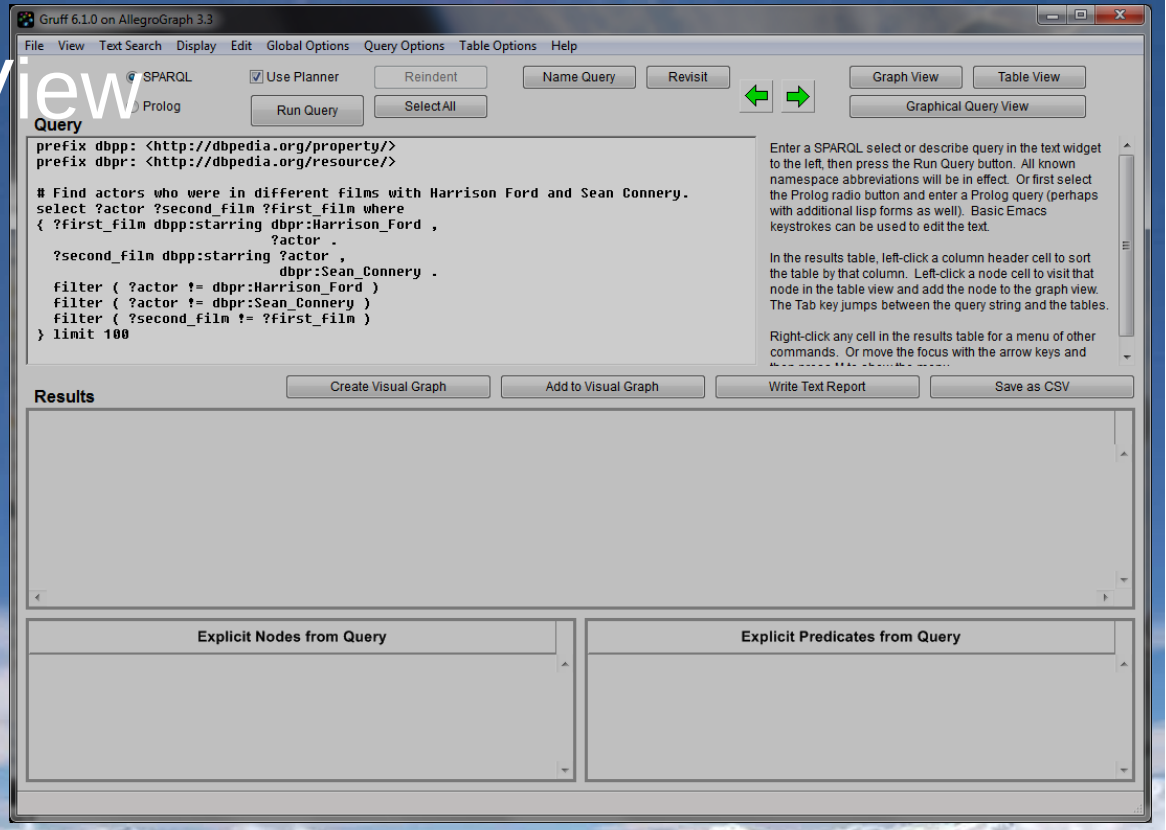
Gruff

- Specify the EP



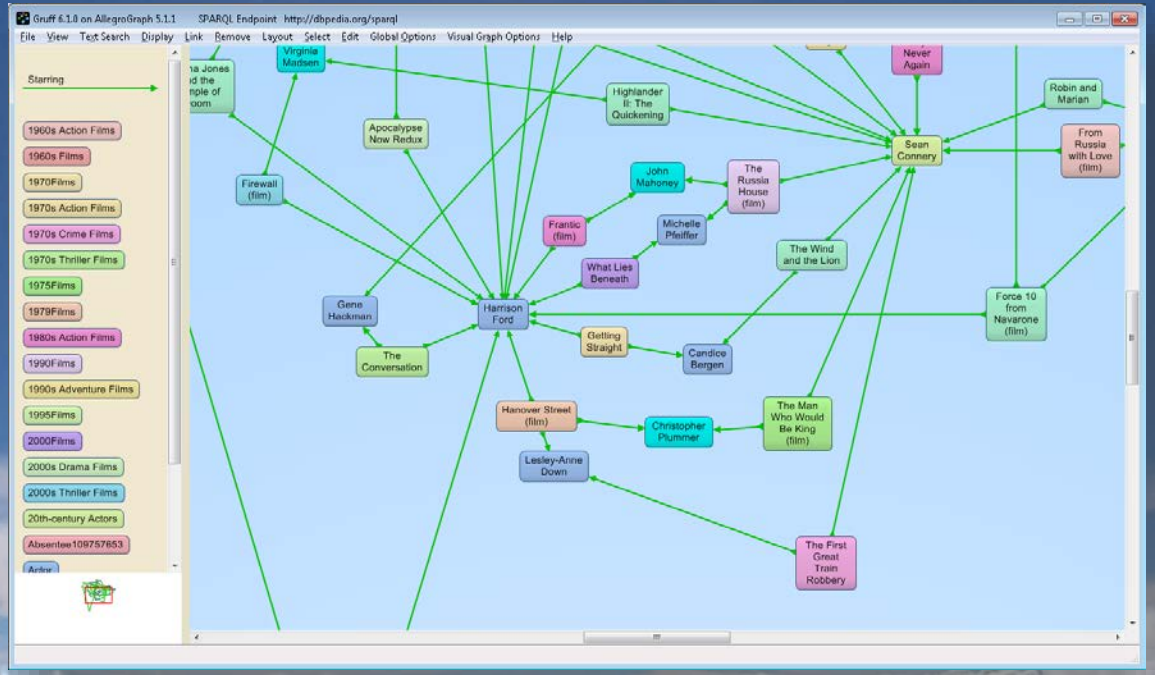
Gruff

- View→Query View



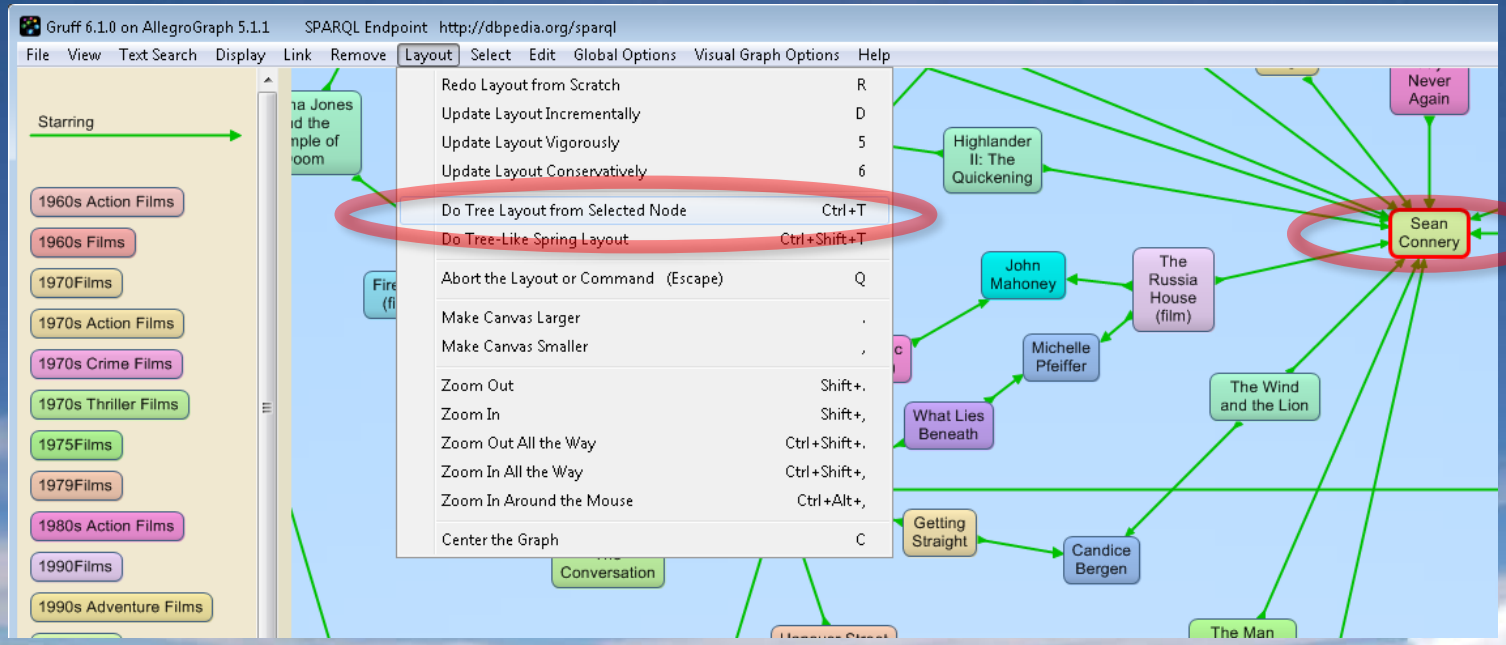
Gruff

- SPARQL query results as a graph



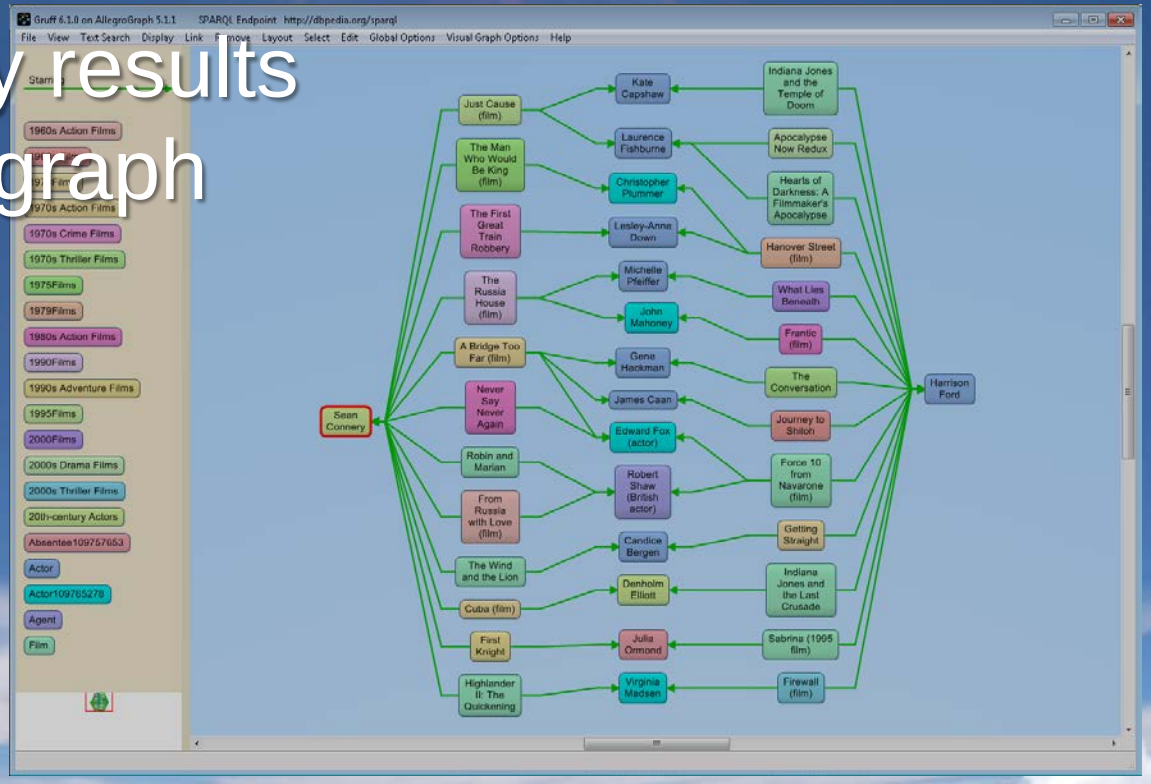
Gruff

- SPARQL query results as a graph



Gruff

- SPARQL query results as an ordered graph



Gruff

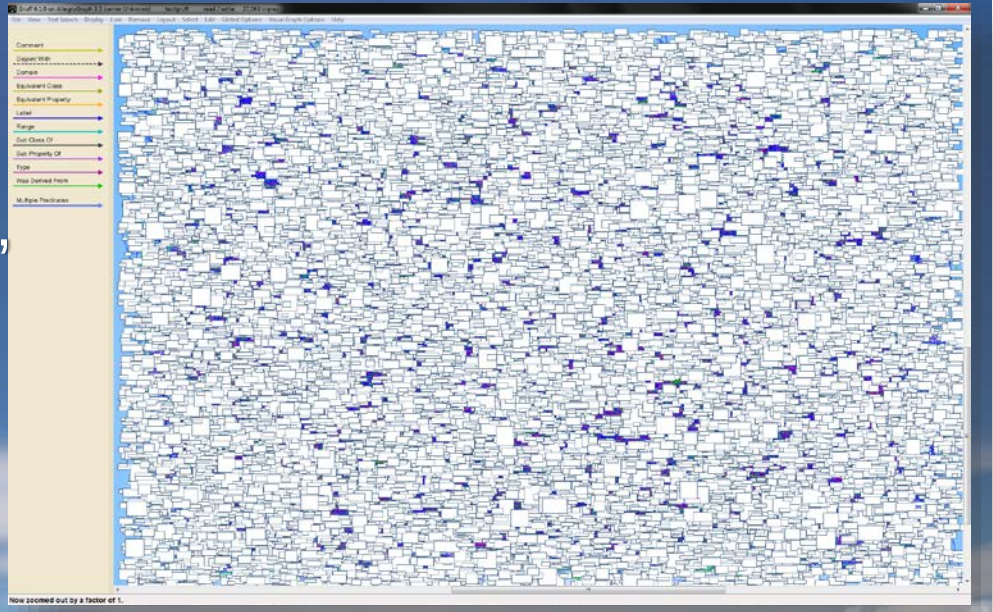
- Benefits
  - Intuitive graph (properties are hidden)
  - Nodes = Resources
  - Easy selection of nodes for a given type
- Drawbacks
  - Limited set of layouts
  - Does not work fine for large datasets
    - That is why, by default, limits the triples to display to 100
      - You can change this limit: Visual Graph Options → Inclusion Options → Maximun Samples Triple to Display (e.g. dbpedia_2014.owl has 27.063 triples)

Gruff

- Trying to visualize the DBpedia ontology
  - Create a new triplestore
    - Select an empty folder in your file system
  - Load RDF/XML (select `dbpedia_2014.owl`)
  - Show graph (by default 100 triples)
  - Remove literals (right-click yellow box on the left → remove)
  - Increase the triple limit to 30,000
  - And….

Gruff

- Trying to visualize the DBpedia ontology
  - And…
  - Wait almost 1 **hour** ☹ later you get an almost "impossible to manage" graph ☹

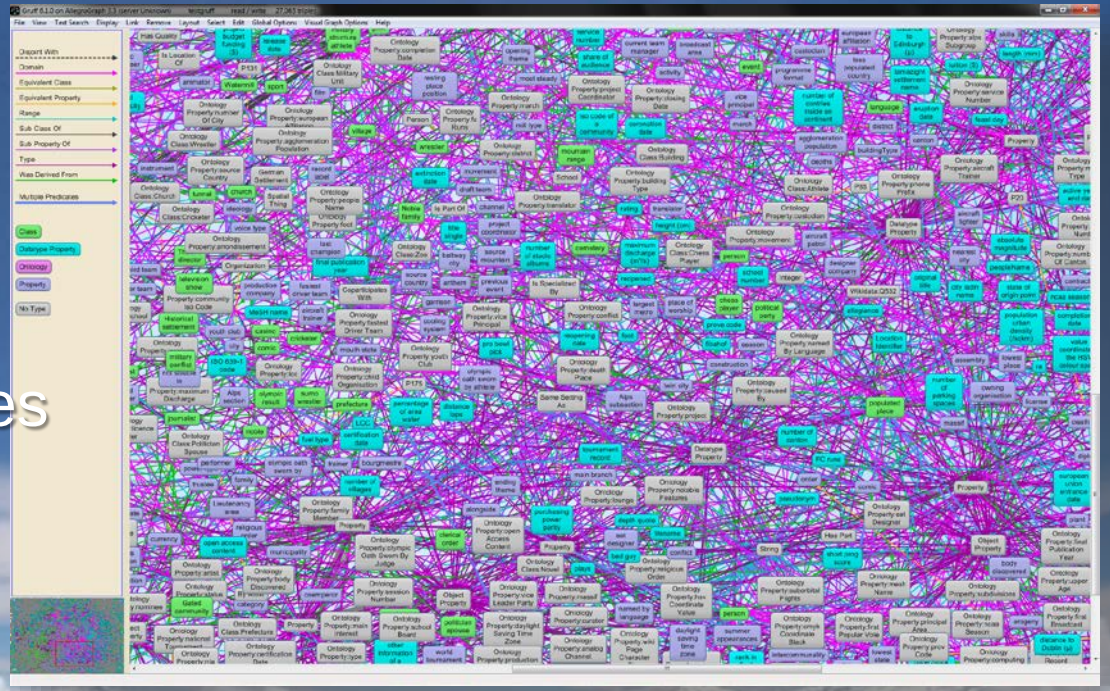


Gruff

- Trying to visualize the DBpedia ontology
  - Zoom in
  - Remove again Literals
    - Wait for hours ☹ again
  - And…

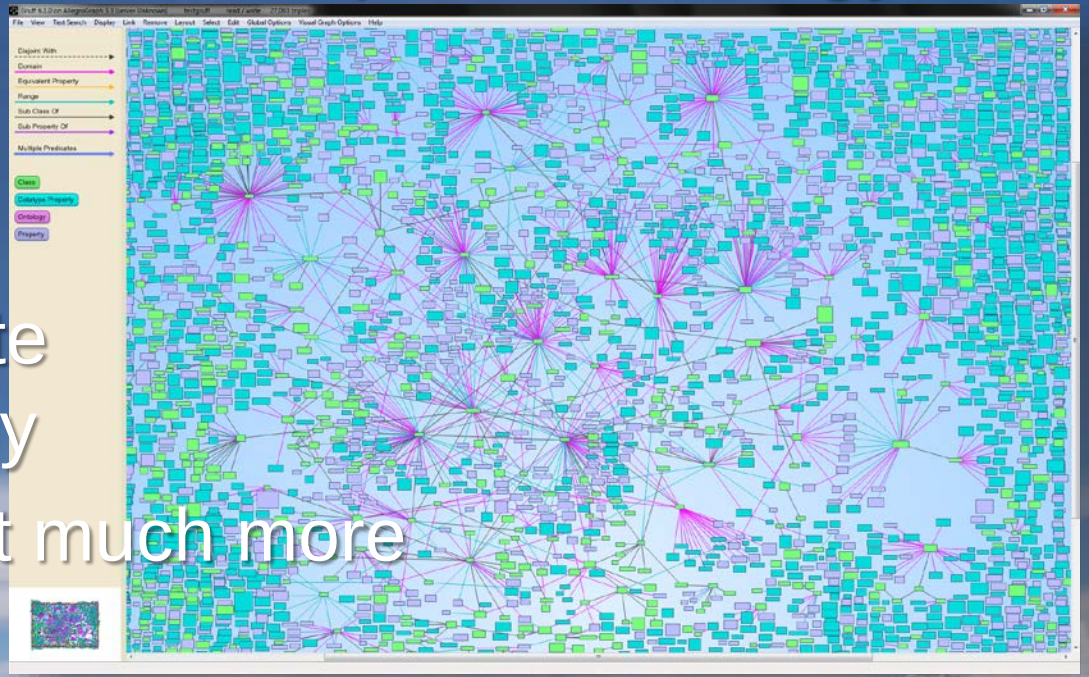
Gruff

- Trying to visualize the DBpedia ontology
  - And…
  - You get this
  - Keep removing
    - Now remove “No Type” nodes
  - And wait



Gruff

- Trying to visualize the DBpedia ontology
  - And…
  - After 20 min…
  - Now, apply Layout→Udapate layout vigosously
  - This is a dataset much more usable



Homework

- Your own proposals
- My proposals
  - Social graph (e.g. Twitter (“TwitterStreamingImporter”))
    - Nodes: persons
    - Arcs: friend of (unlabeled)
  - (DBpedia) ontology graph
    - Nodes: classes. Size ~ instances
    - Arcs (labeled): properties with domain & range
      - How could you manage a “arc size”? (in this case, arc size is proportional to the number of triples using that property)
  - Covid-19 Dataset

Homework

- A written work (pdf) made by up to 4 people. Describe clearly the work done by each author.
- Deadline: April 14th 2025
- Plagiarism is severely prosecuted. Be clever, do not cheat!
- Procedure
  - Upload it to Moodle and
  - Send an email to mariano.rico@upm.es with
    - Subject: DSS Big Data Visualization
    - A pdf file, up to 4 pages, including at least:
      - The SPARQL query (if you use the SemanticWeb Importer) and the rationale behind the SPARQL query
      - The final graph (otherwise agreed, made with Gephi)
      - A description of the operations to achieve the final graph (reproducible results)
      - An analysis of the resulting graph
      - Do not forget to include author(s) name(s)!!.

#VizLinkedData

#VizLinkedData

Thanks for your attention

Mariano.Rico@upm.es

Ontology Engineering Group