## A practical guide to tools which helps you “see” the network



Photo by [Scott Webb](https://unsplash.com/@scottwebb?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText) on [Unsplash](https://unsplash.com/s/photos/network?utm_source=unsplash&utm_medium=referral&utm_content=creditCopyText)

*Every code from this article is published in this* [*repository*](https://github.com/imohitmayank/got_network_viz_python)*.*

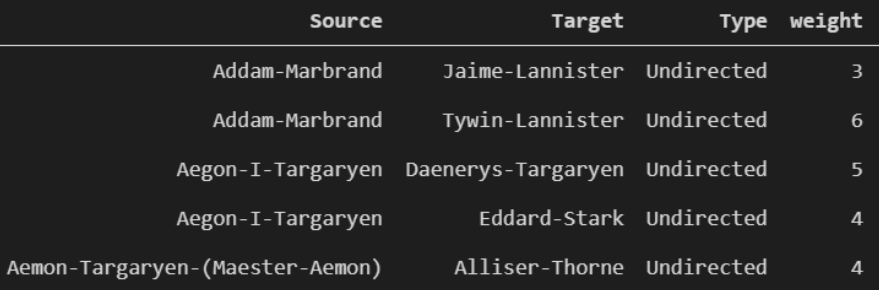
**Update 2nd Feb, 2021:** I recently released [Jaal](https://github.com/imohitmayank/jaal), a python package for network visualization. It can be thought of as the 4th option in the list discussed below. Do give it try. For more details, see [this separate blog](https://mohitmayank.medium.com/introducing-jaal-interacting-with-network-made-easy-124173bb4fa). Thnx!

## Introduction

Network or Graph is a special representation of entities which have relationships among themselves. It is made up of a collection of two generic objects — (1) node: which represents an entity, and (2) edge: which represents the connection between any two nodes. In a complex network, we also have attributes or features associated with each node and edge. For example, a person represented as a node may have attributes like age, gender, salary, etc. Similarly, an edge between two persons which represents ‘friend’ connection may have attributes like friends\_since, last\_meeting, etc. Because of this complex nature, it becomes imperative that we present a network intuitively, such that it showcases as much information as possible. To do so we first need to get acquainted with the different available tools, and that’s the topic of this article i.e. to go through the different options which help us visualize a network. Let’s get started!

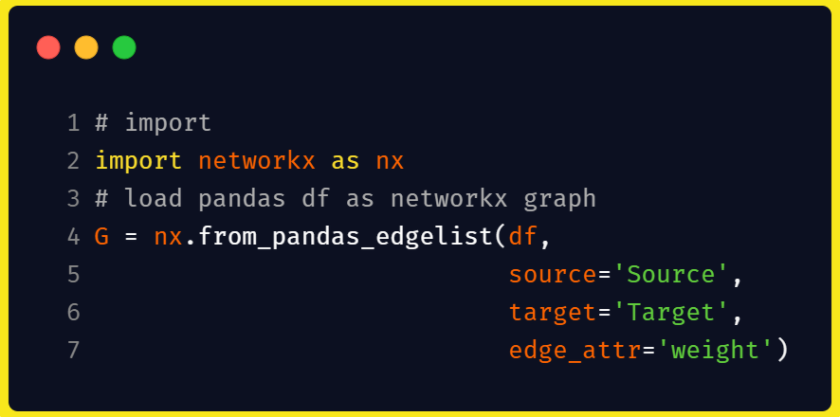
## Network analysis using NetworkX

Before diving into visualizations, let us first understand how does a graph data look like and how can we load it into memory using [NetworkX](https://networkx.org/) in Python. Below we can see the tabular formulation of Game of thrones social network. Here, nodes represent the characters of GoT and edge between two characters means that their names co-occur in the vicinity of 15 words from one another in the books.



The first two columns contain the nodes (here the GoT characters), and one pair of Source and Target represents an edge between the two characters. Only looking at the dataset of book 1 we have 187 unique characters and 684 connections (rows). Also, the weight column gives a sense of the importance of the connection, here its the number of times we have seen the names of two characters in the vicinity (as defined above) in book 1. To make the network manageable, we only consider the strong connections by keeping edges with weights>10. This trims down the graph to 80 nodes (characters) and 175 edges (connections).

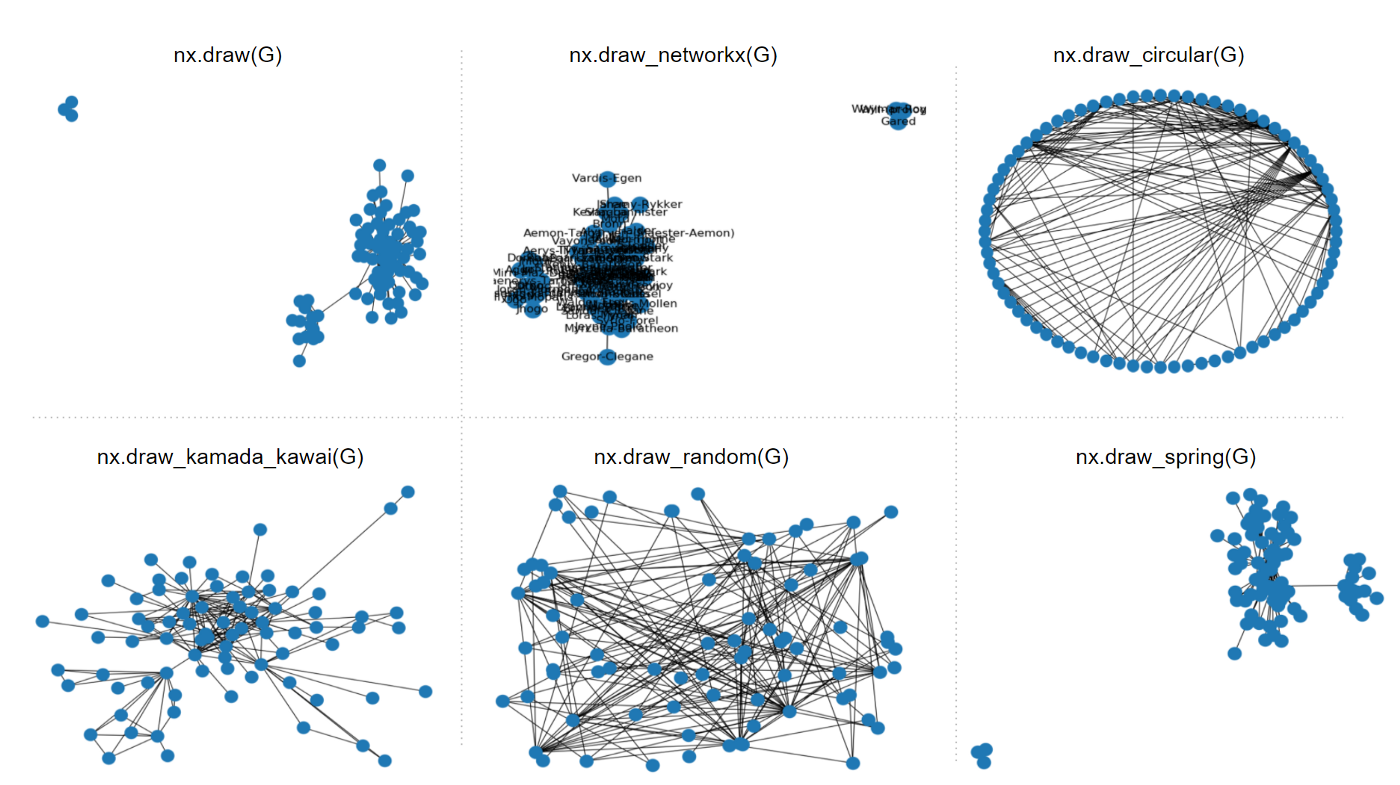
Loading this data, which is in pandas dataframe format, as a network is quite easy. It can be done using NetworkX as follows,



And that’s it! The variable G is now a networkx graph on which we can perform graph-related operations. Now, done with the pre-requisite, let explore different visualization options one by one.

## Option 1: NetworkX

NetworkX has its own [drawing](https://networkx.org/documentation/stable//reference/drawing.html) module which provides multiple options for plotting. Below we can find the visualization for some of the draw modules in the package. Using any of them is fairly easy, as all you need to do is call the module and pass the G graph variable and the package does the rest.

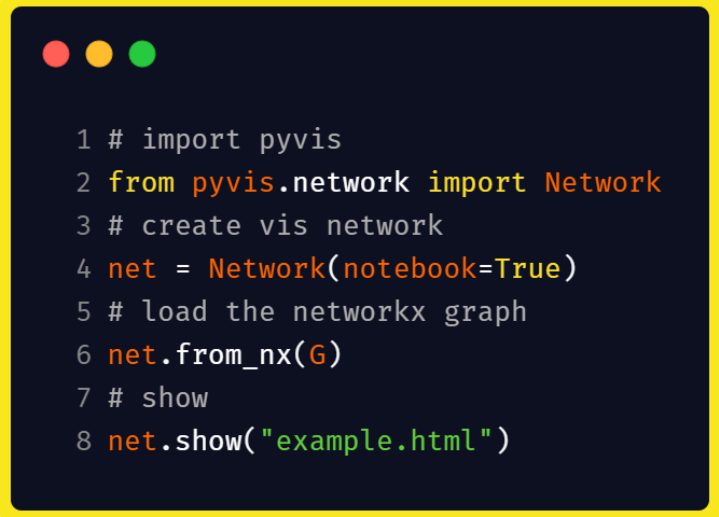


While the visualization option is built in the default python graph package and is quite easy to call, it’s highly counter-intuitive and good only for small networks. Most of the time, with large networks, any of the inbuilt module calls doesn’t make a lot of sense. This makes the default option not the obvious choice if you are using larger network data. Another downside, it’s not interactive so the plot is a fixed graph. This is a major drawback as there are other options which let you manually interact and play with the graph. With this cue, let’s move on to our next option.

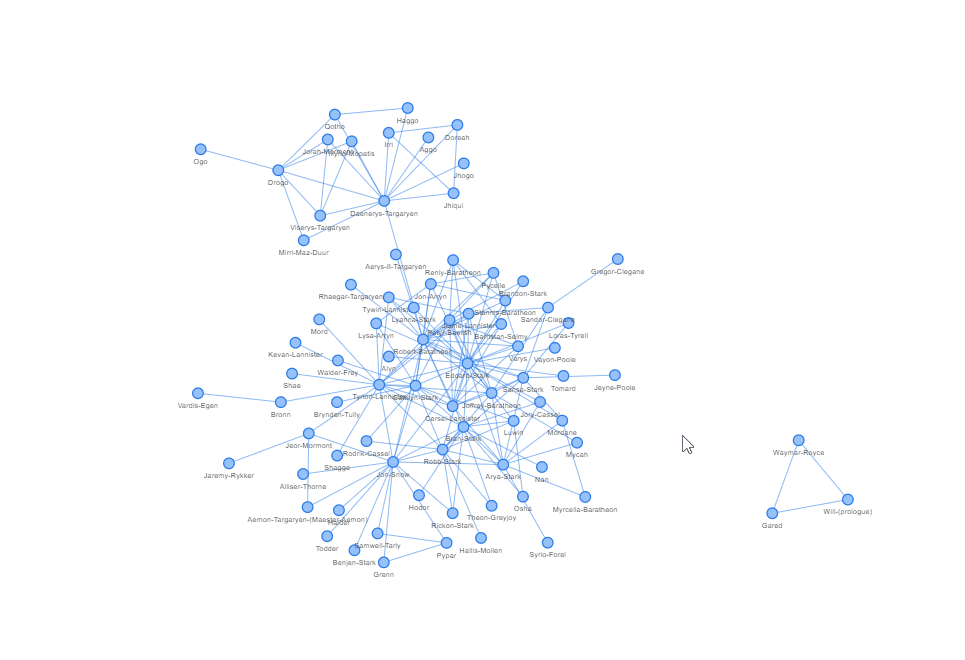
## Option 2: PyVis

[PyVis](https://pyvis.readthedocs.io/en/latest/index.html) is an interactive network visualization python package which takes the NetworkX graph as input. It also provides [multiple styling options](https://pyvis.readthedocs.io/en/latest/tutorial.html) to customize the nodes, edges and even the complete layout. And the best part, it can be done on-the-go using a setting pane where you can play with the various options and export the final settings in form of a python dictionary. This dictionary can later be passed as config while calling the function, resulting in as-it-was drawing of the network. Apart from this, in terms of visualization, you have the basic option of zooming, selecting, hover, among others. Cool isn’t it! 😉

By default, drawing our GoT network can be done easily by,



This plots the network as,

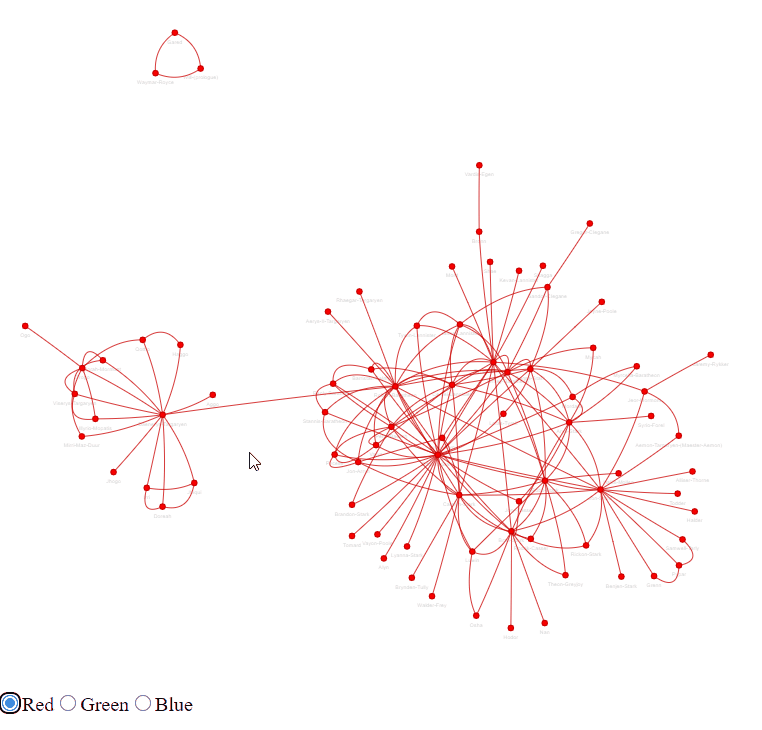


## Option 3: Visdcc in Dash

One major drawback of previous options is that they are very difficult to use with interactive dashboards like [Dash](https://plotly.com/dash/). This is so because apart from supporting manual interactions like select, zoom, etc, a package should automatically adjust over programmatical interactions like change in data, change in properties, etc. This feature is supported by [visdcc](https://github.com/jimmybow/visdcc) which is a port of [visjs](https://github.com/visjs/vis-network) in Python. This makes it fairly easy to modify the graph or even some select properties of the graph by callbacks, which in Dash can be connected to widgets like buttons or radio select options. A sample Dash app for our GoT dataset is shown below,



Here, the code plots the GoT network as we did previously. Apart from that, we have added a callback on the graph, such that on select of an option we change the colour of the complete graph. Note this is a dummy example, so the complete scope is quite immense like adding search options (find any one character), tune the filter on weights (moving from our fixed value of 10), etc. The Dash app with the dummy example is shown below,



## Conclusion

The intention of this article was to introduce the reader with network data and the different options in Python for its visualization. Our list of options started with an inbuilt NetworkX plotting module, which can be used to visualize small and non-complex (fewer connections) graphs. For larger graphs, we can use PyVis as it supports auto-layout (forcing the nodes to be as apart as possible) and provides manual interactions (zoom, drag, select, etc). Finally, in extreme cases where we want to further play with the network by analyzing the change in network w.r.t. nodes and edges properties, we can use visdcc to plot network in Dash and connect the features to graph with callbacks. Hope this was of help 🙂

Cheers.