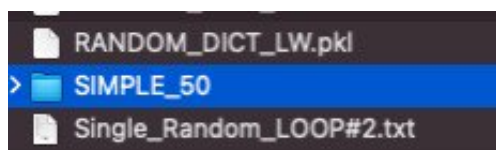


# 网络文件使用说明

## 1. 生成网络

---

先创建一个文件夹，最好标注好网络的类型和节点数。



把 网络生成.zip 里面包含的三个文件：

- Document\_Save.py
- Gen\_DATA.py
- class\_network\_1.py

放到该文件夹中，打开 Gen\_DATA.py

```
NT_DICT = {'1': {'name': nx.barabasi_albert_graph, 'arg': {'n': 100, 'm': 5}},
           '2': {'name': nx.watts_strogatz_graph, 'arg': {'n': 100, 'k': 10, 'p': 0.5}},
           '3': {'name': nx.barabasi_albert_graph, 'arg': {'n': 100, 'm': 7}}

DOCUMENT_NAME = "SMALL_50" #根据网络结构自行定义，建议带上节点数，网络类型

N_LAYERS = 3

N_NODES = 100
```

编辑方法同之前类似，层数 N\_LAYERS 和节点数 N\_NODES 记得也要更改。

DOCUMENT\_NAME 改成和文件夹一样的名字

然后便可以得到网络。



网络生成.zip	今天 下午 1:04
class_network_1.py	2020年12月21日 下午 7
detail.txt	今天 上午 11:38
Document_Save.py	2021年11月13日 下午 8
Gen_DATA.py	前天 上午 8:32
SMALL_50_Given_0.1	今天 上午 11:39
SMALL_50_Given_0.2	今天 上午 11:40
SMALL_50_Given_0.3	今天 上午 11:41
SMALL_50_Given_0.4	今天 上午 11:42
SMALL_50_Given_0.5	今天 上午 11:43
SMALL_50_Given_0.6	今天 上午 11:44
SMALL_50_Given_0.7	今天 上午 11:45
SMALL_50_Given_0.8	今天 上午 11:46
SMALL_50_Given_0.9	今天 上午 11:47
SMALL_50_Random	今天 上午 11:47

子文件夹中包含3个文件：

SMALL_50_Given_0.8_Compress_NT.nt
SMALL_50_Given_0.8_SP_Info.pkl
SMALL_50_Given_0.8_Wei_Btw_Layer.pkl

- 后缀 `_SP_Info.pkl` 记录跟最短路权重有关的信息
- 后缀 `Wei_Btw_Layer` 记录节点的权重

## Step.1大概测试

`SPREAD_ALL.py` 请放在文件根目录下

`SPREAD_ALL.py`：打开每个文件夹，读取网络，传播一次，生成详细传播的csv文件。

需要设置的参数：

`RADIUS` , `BETA`

根据不同的选取策略需要更改从模块 `MULTI_SPREAD` 选取的函数

## Step.2 单个网络多次测试

如果需要对某一个个顶文件夹进行多次重复，需要用到 `For_One_Network.py`

将该文件放到某个文件夹中，设定要循环次数，运行即可。

```
For_One_Network.py
SMALL_50_Given_0.3_Compress_NT.nt
SMALL_50_Given_0.3_SP_Info.pkl
SMALL_50_Given_0.3_Wei_Btw_Layer.pkl
SMALL_50_Given_0.3.csv
SMALL_50_Given_0.32021_11_19_20_10_04.csv
SMALL_50_Given_0.32021_11_19_20_12_03.csv
```

## 关于ForOneNetwork.py

需要设定的有:

BETA , RADIUS ,

REPEAT\_TIME :循环的重复次数

## 关于选层选种文件NodeLayerSel.py

-

`Node_Sel_Betw(MULTI_NETWORK, N_LAYERS, N_NODES, wei_dis_dict, RADIUS)`

使用介数中心性

`Node_Sel_Neib(MULTI_NETWORK, N_LAYERS, N_NODES, wei_dis_dict, RADIUS)`

使用带半径的邻居数 `R_LAYER_R_NODE(N_LAYERS, N_NODES)` 随机层, 随机节点

`R_LAYER_Degr_NODE(MULTI_NETWORK, N_LAYERS, N_NODES)` 随机层, 节点用度选

取 `R_LAYER_Neigh_NODE(MULTI_NETWORK, N_LAYERS, N_NODES)` 随机层, 节点用邻居选取

## 2021.12.14 现已采用更灵活的方式

已经将选取层和选取节点分开

默认情况下, `wei_dis_dict` 在主程序中为 `SP_Info`

## 关于层的选取:

`R` 代表随机, `Betw` 代表用介数中心性进行选取, `Neib` 表示用

```
Betw_Layer(MULTI_NETWORK, N_NODES, N_LAYERS, wei_dis_dict)
```

```
R_Layer(N_LAYERS)
```

```
Nei_Layer(MULTI_NETWORK, N_LAYERS, N_NODES, wei_dis_dict, RADIUS)
```

## 关于节点的选取：

`R` 代表随机，`Degre` 表示用度来选取，`Neigh` 表示用邻居来选取，`Gravity` 表示用引力模型

`Top` 参数表示选取排名的前**Top**个节点，不填默认选第一个

```
R_Node(N_NODES, Top = 1)#随机选Top个
Neigh_Node(MULTI_NETWORK, N_LAYERS, N_NODES, Top = 1)
Degr_Node(MULTI_NETWORK, N_LAYERS, N_NODES, Top = 1)
Gravity_Node(MULTI_NETWORK, wei_dis_dict, Pick_Layer, N_LAYERS, N_NODES, RADIUS)
```

## 例子

```
from Node_Layer_Sel import Betw_Layer#选层
from Node_Layer_Sel import Gravity_Node#选点
...
Max_Layer = Betw_Layer(MULTI_NETWORK, N_NODES, N_LAYERS, SP_Info)
Max_Node = Gravity_Node(MULTI_NETWORK, SP_Info, Max_Layer, N_LAYERS, N_NODES, RADIUS)
```