Configuration model (CM)

The configuration model allows the generation of random networks having any desired degree distribution, and it is also used for the randomization (through rewiring) of any given network.

The input of the configuration model algorithm is the degree distribution of the nodes, and the output is a complex network whose nodes follow the desired degree distribution; apart from this constraint, the rest of the structure of the network is random. The steps of the algorithm are the following:

1. Select the size *N* of the network (the number of nodes).
2. Generate *N* random numbers according to the input degree distribution, which will become the degrees of each node.
3. Assign to each node as many "stubs" as indicates its degree (1 stub = 1/2 edge).
4. While there are free stubs:
    a. Select one stub.
    b. Randomly select another stub.
    c. Connect the previous stubs to form an edge.

There are some details to take into account for a successful implementation of CM:

- The total number of stubs must be even, otherwise one stub will be free; just regenerate the degree of any node.
- Suppose we have 5 nodes, with degrees assigned in step 2 equal to 5, 3, 2, 4 and 6, and that all stubs are free.
- In step 4a, we can choose the first free stub, in this example the first stub of the first node.
- In step 4b, the random selection of a stub is equivalent to generate a uniform random integer between 1 and 20 (=5+3+2+4+6); e.g., if we get 12, the selected stub is the second stub of the fourth node. We cannot choose randomly first the node (with a random number between 1 and 5) and then a free stub of that node, since the probabilities of the stubs would become non-uniform, producing a bias in the resulting (incorrect) network.
- In step 4c, the connected stubs are removed from the list of free stubs, remaining in the example 18 (=4+3+2+3+6) free stubs.
- During the process of connecting stubs, you must avoid the appearance of multiple edges (several edges between the same pair of nodes) and self-loops (and each from one node to itself); basically, generate a new random number whenever this happens.
- When the number of remaining free stubs is small, it may be difficult (even impossible) to avoid multiple links or self-loops; apply reasonable strategies to

cope with them, e.g. remove stubs, undo some links, etc., always trying to maintain the desired degree distribution and random assignment of edges.

- If the average degree corresponding to the desired degree distribution is small, it is possible that the resulting network could be formed by several disconnected components. Just be aware of this fact and, if necessary, take the necessary actions to avoid it.

An alternative to the previous algorithm is the following:

1. Select the size *N* of the network (the number of nodes).
2. Generate *N* random numbers according to the input degree distribution, which will become the degrees of each node.
3. Assign to each node as many "stubs" as indicates its degree (1 stub = 1/2 edge).
4. Create a vector of stubs: *v*[*s*] = *i* (i.e. stub in position *s* belongs to node *i*)
5. Find a random permutation of the vector *v* of stubs, e.g. using the algorithms of Fisher-Yates, Durstenfeld or Knuth.
6. Connect consecutive stubs: (*v*[1], *v*[2]), (*v*[3], *v*[4]), (*v*[5], *v*[6]), etc.

Again, we must control the appearance of multiple edges and self-loops.