Supplementary Material

ODD for “BC Robustness” model, associated with the article “Testing the robustness of local network metrics in research on archaeological local transport networks”

Mark R. Groenhuijzen, Philip Verhagen

# Overview

## Purpose

The purpose of this model is to measure the development of the network measure of betweenness centrality of a selected node under the influence of randomly adding nodes and their associated connections to the network from a previously created dataset (thus all runs converge to the same outcome). The model was developed with the objective to test the robustness of betweenness centrality, which is generally measured in the final or ‘complete’ state of the network, but which can be susceptible to minor variations (e.g. due to uncertainties in the dataset) in the network analyzed.

## Entities, state variables, and scales

This model was developed for an archaeological case study: a transport network in the Dutch part of the Roman *limes* during the Middle Roman Period (~70-270 AD). The principal agents are archaeological sites, containing some Roman forts (*castella*) and other military sites, but chiefly consisting of small rural settlements. These sites are loaded from an archaeological site database constructed within the project (Verhagen et al. in prep.). Within the model, the individual sites are identified with a unique ID (‘LimesLimitID’). During a model run, the sites’ betweenness centralities are continuously recalculated as new sites are added to the network. These state variables are essential for a complete model output. The sites also have other parameters in the model, such as site type and geographic location, but they are not relevant for the outcome of the model and serve an aesthetic purpose.

The sites are connected with links, representing a transport network. The transport connections were modelled in previous research (Groenhuijzen and Verhagen 2015a; Groenhuijzen and Verhagen 2015b) and are loaded during the setup of the model. Four different modes of transport can be used in the model: regular walking, walking while carrying a heavy load, mule-cart transport and ox-cart transport. A link in the model is identified with the unique ID’s of the source and destination sites. Furthermore, a link contains a travel time parameter (measured in seconds), which can be used to filter the dataset to include only connections that can be traveled within a certain time limit (by default set to 3600s).

The model utilizes a spatial framework to locate the sites, defined in the WGS84 coordinate system. Sites are positioned according to the location read from the archaeological site database. However, this has no influence on the outcome of the model.

## Process overview and scheduling

The model starts with the setup of the relevant datasets: the archaeological sites and the transport networks are loaded from source files. A ‘key site’ then has to be selected for which betweenness centrality and the related parameter of betweenness rank are tracked during this model run. All other sites and connections are subsequently made ‘absent’ from the network, so that they are not taken into account in the recalculation of the network measures.

The main process of the model starts by adding five sites picked randomly from the previously loaded dataset, along with all associated links (as long as the sites on both ends of the link are ‘present’). Betweenness centrality is subsequently recalculated (see section “Submodels”), and the measures of the ‘key site’ are recorded for output. This process is repeated until all sites have been added to the network.

The model has been prepared for use in NetLogo’s BehaviorSpace module, which allows for multiple parallel runs. By default it has been set up to run 100 times for each site, and to run through the entire site dataset.

# Design concepts

This model utilizes concepts of network science, as the objective is to test the development of a network measure under the influence of randomly adding sites to the network. The model’s focus is on the local network measure of betweenness centrality, which is a measure of the amount of shortest paths between all pairs of other nodes that pass through a node (Freeman 1977). The model also measures betweenness rank, which in this research is defined as the measure of a site’s betweenness centrality in relation to all other sites: the site with the highest betweenness centrality is given a value of 1 and the site with the lowest betweenness centrality a value equal to the number of sites present. Additionally, a distinction is made between absolute betweenness rank (i.e. the rank in relation to the total number of other sites in the network) and the percentage betweenness rank (i.e. the percentage of measured sites that have a higher betweenness centrality). The latter is used in the research as a normalization of betweenness centrality, since betweenness centrality is also dependent on the number of sites present.

# Details

## Initialization

After loading the source data, the model starts with only the ‘key site’ being set as ‘present’, i.e. the site for which betweenness centrality and the related parameter of betweenness rank are tracked during the model run. All other sites and associated links are set as ‘absent’. As long as a site or link is absent it is not taken into account in the network for recalculating network measures.

## Input data

The model uses an archaeological site dataset and can use four different modeled transport networks. The input data is static, meaning that they are loaded during the model setup and are not subject to change during the course of a model run or between individual model runs.

## Submodels

The main process in the model is the recalculation of betweenness centrality after the addition of new sites and links to the network. This is performed using the “betweenness-centrality” function in NetLogo’s network extension. This function uses an algorithm from the JUNG software library (White and Nelson 2009) and is based on the algorithm proposed by Brandes (2001). Since the function only produces the amount of shortest paths passing through a node as output, it is normalized by dividing by the total amount of node pairs present.