sDNA Open Documentation

Release 4.2.0

Crispin Cooper

2021

Contents

This is the manual for sDNA Open, the spatial network analysis toolbox for GIS and Python.

This manual can be cited as follows:

> Cooper, C. (2021) Spatial Design Network Analysis (sDNA) Open version 4.2 Manual. Cardiff University.

Please also remember to cite the software itself in any published sDNA analysis:

> Cooper, C.H.V., Chiaradia, A.J.F., 2020. sDNA: 3-d spatial network analysis for GIS, CAD, Command Line & Python. SoftwareX 12, 100525. https://doi.org/10.1016/j.softx.2020.100525

How to get help

Please ask questions on GIS Stack Exchange and tag them with *sdna*. If you get no response within a few days, feel free to chase up the sDNA developers by sending an email to sdna@cardiff.ac.uk with a link to your stack exchange question.

Bugs, Feature Requests, Errata

Please report bugs, feature requests and any errors found in the manual to the bug issue tracker on sDNA Open's github page.

Acknowledgements: This manual was originally put together under an Impact Accelerator grant from the Economic and Social Research Council. The principal investigator was Alain Chiaradia and the co-investigator Crispin Cooper, based at Cardiff University department of Geography and Planning and the Sustainable Places Research Institute.

CHAPTER 1

Table of Contents

1.1 Introduction

1.1.1 What sDNA is used for

sDNA stands for Spatial Design Network Analysis, which does what it says on the tin. Let's take that one step at a time.

- ***Network Analysis*** A network is a structure defined by nodes, and links between them. Many networks exist in this world: virtual, physical, social, and a huge literature exists on analysing networks.
- ***Spatial Network Analysis*** We define a spatial network to be a network with added spatial information. Nodes have positions in Euclidean two- or three-dimensional space. The links between nodes need not take the most direct route, but can have physical shapes other than straight lines. In sDNA, we focus on analysing links, rather than nodes. Links can have data attached to them.
- ***Spatial Design Network Analysis*** sDNA can be used to analyze any kind of spatial network.

  However (... with the exception of one escapade the author had with fish in rivers...) our research focuses on urban networks and transport systems: networks for vehicles, pedestrians (indoors and out), cyclists and public transport. sDNA spans the gaps between transport planning, urban design and architecture.

  Measures output from sDNA have been shown to correlate with a wide variety of phenomena including health, community cohesion, land values, town centre vitality, land use, pedestrian, cyclist and vehicle flows, emissions, accidents and crime. sDNA's high spatial resolution, low data collection cost and applicability to economic, social and environmental problems, makes it particularly suitable for modelling sustainable transport systems; while its ability to handle large networks also makes it suitable for "big data" type analysis.

  So, the "Design" part of the sDNA name is there because our aim as researchers is to provide an evidence base for the design of better networks in the built environment.

This leaves me in the awkward position of writing a manual for two audiences! Firstly, the built environment planners, architects, designers and modellers, who want specific details on modelling their system. Secondly, the people modelling other kinds of spatial network, who want to understand the more abstract and mathematical concepts. I have

tried to cater to both in this manual; if you are in one group, please feel free to ignore any material which is clearly pitched at the other.

Anyone who is using real world network data will need to learn about the potential pitfalls by reading *Network Preparation*, and anyone using sDNA will need to read *Installation and first usage*. It is then possible to start using sDNA without any more reading, though *Guide to individual tools* exists for a reference if needed. For a quick introduction to the concepts behind sDNA there is *Analysis: what the results mean*, and for the full details, *Analysis: full specification*.

Transport, planning and urban design practitioners should start with *Network Preparation* then *Installation and first usage*. If you need to get a job done quickly then proceed to *Step by step guides for specific tasks*; for more in-depth understanding the other chapters await.

1.1.2 Research Evidence

We are currently keeping our list of publications on sDNA on the about page of our website.

1.2 Network Preparation

The phrase "garbage in, garbage out" applies to spatial network analysis just as much as it applies to all modelling. Careful preparation of spatial networks for analysis is essential to getting a meaningful result.

This chapter is a much needed guide to the pitfalls we have found in preparing data for spatial network analysis. Hopefully by following it you can avoid the mistakes we used to make, and have seen others make. Although this guide now forms part of the sDNA manual, it applies equally well to any other form of spatial network analysis.

We present the material as follows

- *Principles* discusses the types of errors that appear in spatial networks and how, in theory, to correct them
- *Tools* discusses specific tools that can be used to achieve this correction, depending on your working environment
- *Notes on specific formats* contains notes on preparing some commonly used network formats

1.2.1 Principles

The relationship between data quality, and analysis quality, is not linear. A single error in a critical place could invalidate an entire analysis; meanwhile, numerous errors in non-critical places can have negligible effect.

Three classes of error should be treated as a matter of priority in spatial network analysis:

- *Data in a misleading format*
- *Connectivity errors at key locations*
- *Geometry errors at key locations*

Data in a misleading format

Connectivity rules

It's easy to assume, just because the software appears to successfully load your data, that the data was in the right format; indeed it is easy to presume that if the format was wrong, the data would have failed to load. With spatial network data this is often not the case.

Software will certainly complain if it doesn't recognise a given GIS and CAD file format (like a shapefile, geodatabase or DWG). But each of these formats are restricted to describing two things, *shapes* and *data*. What they don't describe is the *meaning* of those shapes and data. For example, a shape file could contain two lines which cross like this (blobs are used to make it clear where lines start and end):

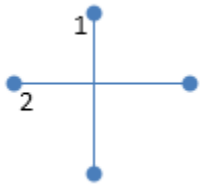

Fig. 1: Figure 4: Two lines in a shapefile, but what do they mean?

As it stands, that's just two intersecting lines. But suppose you knew that the lines represent a street network. Now they have a *meaning* - but the meaning isn't quite clear yet. Are we looking at a crossroads, where two streets join, or a bridge, where one is crossing over another? To answer that question, we need to specify a *connectivity rule* for the data.

sDNA processes spatial networks encoded using a **coincident endpoint connectivity rule.** This is a common standard used in many data sets e.g. Ordnance Survey products as well as other software such as ArcGIS Network Analyst. (Provided you take grade separation into account. Ordnance survey specify grade separations for each end of a link, and links only join if their grade separations match).

The **coincident endpoint connectivity rule**

1. A network is formed by a collection (feature set) of polylines
2. Lines are deemed to be connected if and only if they have coincident endpoints (the lines must end on exactly the same point).

So, as far as sDNA is concerned, the two lines in Figure 1 don't connect, because they have different endpoints. They simply intersect one another, and could either represent a bridge or a tunnel. This category is so common that it has its own name, a *brunnel* (or, in honour of the Victorian engineer, a *brunel*). These features are also called *unlinks*, because the lines represent objects which are not linked.

Figure 1 does not, however, represent a crossroads – and a crossroads is usually what the designer intended. This is a serious data error. The representation of a crossroads, using coincident endpoint connectivity, requires four separate lines:

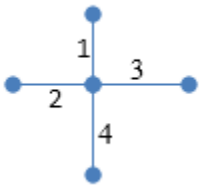

Fig. 2: Figure 5: Four lines in a shapefile, representing a crossroads according to *coincident endpoint connectivity rules*

If the junctions in your network are formatted like Figure 1, not Figure 2, then all network analysis done by software expecting coincident endpoints will be completely meaningless. This cannot be emphasized enough – it may seem obvious, but I have often seen this mistake made in practice, because (unless the software is set to display blobs on the ends of lines, as I have in this document) the two formats both load correctly and look identical to the viewer.

Another reason this mistake is often made is that drawing networks using a coincident endpoint connectivity rule is tedious, time-consuming, painful and error-prone. The designer has to draw a lot more lines than they would otherwise need to, and those lines don't really seem to match up with the way we (as humans) think about space. For these reasons, people don't like to draw networks with coincident endpoint connectivity. This is absolutely fine – avoiding a tedious and error-prone task is to be encouraged. But it is vital that the networks thus produced are converted to coincident endpoint connectivity before analysis.

Of course, if you draw crossroads in the style of Figure 1, how would you then indicate which of these are bridges or tunnels (brunels) rather than crossroads? By using some different connectivity rules:

Alternative connectivity rules

- *Link-Unlink (polygon) rule*. Intersecting links are assumed to join unless they are specifically marked otherwise. This is done with a separate layer of unlinks (drawn as polygons). This is an easy format to draw by hand.
- *Link-Unlink (data) rule*. Intersecting links are assumed to join unless they are specifically marked otherwise. This is done by having a brunel data field attached to each line. This is an easy format to draw by hand.
- *Link-Endpoint Grade Separation rule*. Intersections are not allowed in the data at all; ALL lines where they touch MUST have coincident endpoints. Where links don't join at their endpoints this is shown by providing elevation or grade separation data – two data fields attached to each line, one for the line start and the other for its end. (OS ITN does this).
- *Link-Node rule*. Intersections are not allowed in the data at all; ALL lines where they touch MUST have coincident endpoints. Where links don't join at their endpoints this is shown by providing elevation or grade separation data – endpoints are assumed not to join unless their elevation and grade separation match. This data is provided in a separate layer of points, cross-referenced to the lines whose elevations they represent (OS Meridian does this).
- *Shared point rule* Lines do not intersect unless they share a point in the list of points representing the line in the data. This could be an endpoint, or any corner in the line itself, or even added without a corner to create an intersection. OpenStreetMap uses this rule, though refers to all points within the lines as 'nodes'; in this document I reserve use of 'nodes' for places where links are connected.

Note one advantage of the latter two formats is that intersecting lines are not allowed at all; therefore any detected by software are guaranteed to be data errors. Catching errors in data is an important function we discuss later.

The problem then remains to convert from these other formats to a network with a coincident endpoint connectivity rule.

- **For link-unlink (polygon) rule**, sDNA for Autocad includes a tool to convert the network. If you need this tool for GIS, drop us a line – nobody has asked for it yet but we can look at making it. Alternatively, in any GIS system, convert the network to link-unlink (data) format as follows:
  - use the unlink polygon layer to split links at unlink polygon boundaries
  - create a *brunel* data field on the link layer
  - select links within the polygons of the unlink layer, and mark them as *brunels* in the data field
  - your network is now in link-unlink (data) format. Follow the steps below to convert **link-unlink (data)** into coincident endpoint format.
- **For link-unlink (data) rule**, use a *Break intersections* tool to break all lines except for brunels. Split the layer into two separate layers; one for brunels, the other for the rest of the links. Break intersections on the link layer, then rejoin the layers.

- **For Link-Endpoint Grade separation rule**, there is no need to convert. All sDNA tools accept start- and end-grade separation data attached to links; endpoints will be assumed to connect if and only if their grade separations match.
- **For Link-Node rule**, get in touch with us – although they are not released to the public yet, we have some tools to do this.
- **For Shared point rule** use a *Shared point line breaker*

Summary: ensure you understand spatial network connectivity rules, and (for sDNA) that your network uses coincident endpoint connectivity.

Spatial reference

Returning to Figure 1 and Figure 2, even though we may know what the lines represent – and their connectivity rule – we still don't know what the coordinates represent. This is the *spatial reference* of the data. The issue is important both when creating new data, and when downloading existing data from other sources.

Spatial references can either be Geographic or Projected. Geographic coordinate systems specify a position on the surface of the earth, by measuring angles of latitude and longitude from the earth's axis. WGS84, the dominant global standard for GPS and mapping data, is an example of this.

While geographic coordinate systems describe a curved surface, projected coordinate systems simplify the representation of small areas by making the approximation that they are flat. Each country usually has a national grid to use for this purpose, based on x and y coordinates.

To analyse a network, you probably want to use units of distance that have some physical meaning to you – metres, miles or kilometres. If you have downloaded data that uses a geographic coordinate system, this usually means you will need to convert (project) it using your GIS or CAD software.

Unfortunately, not everybody in the world understands spatial references. This means that sometimes, data you receive may be marked with the wrong spatial reference – watch out!

Another problem which can arise, is that the GIS software you use may not correctly identify the transform needed to project the data you downloaded onto a sensible grid. So, it is best to double check, and manually select the correct transform if needed.

In the case of sDNA, if you are not sure whether your data is projected properly, you can check what sDNA 'sees' as the length of links by running the sDNA Individual Line Measures tool. You can then check what sDNA reports for the length of one or two different links. Is it what you expected? If not, you probably have the wrong spatial reference and will need to reproject your data.

Connectivity errors at key locations

Even though you may have used the correct connectivity rule for your network, it is possible that you or somebody else made a mistake when drawing it. Figure 3 shows the same network as figure 2, at four levels of increasingly greater magnification:

Oh no! What looked like a perfectly normal crossroads from a distance, turned out to be disconnected when we zoomed in. This is potentially a very serious error. If you want an accurate model, this is of course unacceptable – but even if you only want an approximate network model, if such a disconnection is in a key location (such as a major through route in a city) then the entire analysis could be meaningless, even for a rough model.

The following sections deal with this, and other, forms of connectivity error.

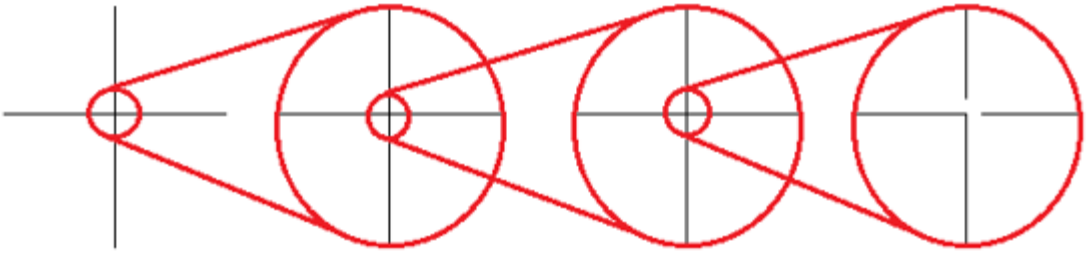

Fig. 3: Figure 6: Zooming in on a faulty crossroads

Cluster tolerance

We first introduce the idea of a *cluster tolerance*. This is a small distance; on an urban scale perhaps 1cm (though larger cluster tolerances can be useful sometimes). If two points are closer than the cluster tolerance, then they are considered to be the same point. sDNA supports two types of cluster tolerance, XYTolerance for horizontal differences between points, and ZTolerance for vertical differences.

Note that some GIS systems, notably ArcGIS, will display unconnected lines as connected – no matter how closely you zoom in – if the disconnection is smaller than the cluster tolerance of the GIS! So, we have to be very careful about whether lines are connected, because we may not even be able to see errors, regardless of how closely we look.

The sDNA Prepare tool supports fixing of disconnections smaller than the cluster tolerance. In the case of Figure 3, if you set a suitable cluster tolerance then sDNA Prepare will connect the lines (provided the network in Figure 3 actually consists of four lines, rather than two – see *Connectivity rules*). sDNA cluster tolerance applies only to the endpoints of lines, not any points in between. In ArcGIS, sDNA uses whatever tolerance ArcGIS is using by default, though a custom tolerance can also be set. On other platforms, the tolerance defaults to zero, i.e. no correction is performed.

But, let's say you have downloaded some network data of unknown quality. How can you tell what cluster tolerance to use? Too small, and you will disconnect things that should be connected; too big, and you will connect things that should be disconnected.

The answer is to use sDNA Prepare to detect (but not fix) cluster tolerance errors.

Fixing a network with unknown cluster tolerance

1. Start with a small tolerance size
2. Use sDNA prepare to detect tolerance problems ("Edge endpoints closer than cluster/XYTolerance"), and examine the errors it finds.
3. Are the features detected all errors, or are some of them genuine small gaps that should not be connected?
    a. If all the features detected are genuine, then the network is ready to use.
    b. If some - but not all - of the features are genuine, choose a smaller tolerance, and go back to step 2.
    c. If all of the features are errors, use sDNA prepare to fix them. Then, choose a larger tolerance and go back to step 2.
    d. If no features are detected at all, then choose a larger tolerance and go back to step 2. If you keep alternating between (b) and (d) above, then the errors and genuine features are the same size. This means the network is of very poor quality and must be fixed by hand.

Intersections and overlaps

Another kind of connectivity error can arise when lines intersect or overlap. Really this is a restatement of *Connectivity rules*, but here we focus on errors that can happen when encoding the endpoint connectivity rule, rather than choosing the wrong rule altogether.

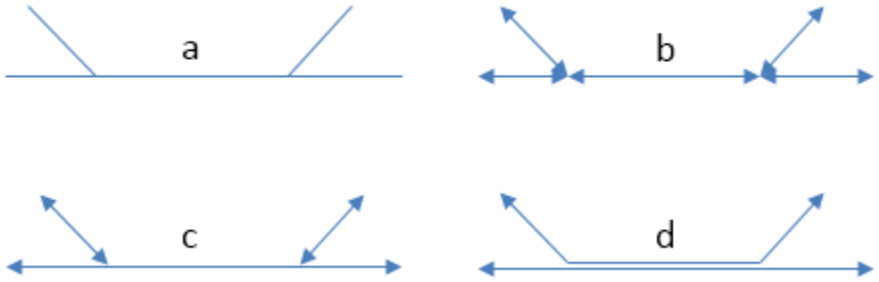

Fig. 4: Figure 7: (a) a pair of junctions; (b) correct set of lines to represent junctions (arrowheads included to show endpoints); (c) incorrect set of lines to represent junctions - intersection error; (d) incorrect set of lines to represent junctions – overlap error. The parallel lines in (d) actually overlap, they are shown here slightly separated for clarity.

Figure 4 shows a set of two junctions (a), represented correctly (b) and also incorrectly (c, d). Note how use of arrowheads at both ends of each line allows you to see errors more clearly – when checking a network, it is recommended practice to configure your GIS to do this[2].

These kinds of errors can be fixed using *Topology* tools. If grade separation data is present, care must be taken that that breaking intersections does not move the grade separation data to the wrong link: it is advisable to search for all nonzero grade separations and check them by hand after line breaking operations.

We recommend breaking links where bridges/tunnels exist, so long as grade separation data is correctly attached to these. This has the advantage that *all intersections become errors*: if you see an intersection, you can be sure it is a mistake in the network rather than a brunel, as the latter would be coded with grade separation data. A similar policy can be applied to *loop links* which start and end at the same point (such as a circular cul-de-sac). If these are deliberately broken into two links rather than one, you can be certain that all loop links are errors also. This prevents the drawing of e.g. a roundabout that is not connected to anything.

Incorrect grade separation, elevation and oneway data

A third kind of connectivity error arises when grade separation, elevation or one-way information is incorrect. In the former cases, even overlapping endpoints will be wrongly assumed disconnected if the grade separation or elevation data is wrong. In the latter case, a one-way street pointing the wrong way can lead to an impossible situation.

The way to check this kind of information is to display it and have a look. If you have grade separation data, it will take the form of two data fields on each link – one for each end of the link (in sDNA, these are usually called start_gs and end_gs). The start and end of a line refer to the way it was drawn – note this is *not necessarily the same direction as the flow of a one-way link.* So, for start and end grade separation to be interpretable, the GIS should be set to display arrowheads only at the end of lines, to make clear which end the start and end grade separation refer to.

In ArcGIS, you can get custom labelling of link start and end grade separation from the *Layer properties -> label features* dialog. Turn on labelling and in *Placement properties* choose *Place one label per feature.* Click on *Expres-*

[2] In ArcGIS the arrow settings are rather well hidden. Look at *layer properties -> symbology -> symbol -> edit symbol -> set type to "cartographic line symbol" -> line properties*.

sion to create a label from multiple fields. The following is an example of an expression that labels each link with connectivity, start and end grade separations; putting the grade separations in brackets for easier reading:

```
[LConn] & " (" & [start\_gs] & "/" & [end\_gs] & ")"
```

(For more information on computing and checking connectivity, see the following section).

To display one-way data, it is best to show an arrow in the direction of the one-way street. In ArcGIS this can be achieved using a 'categories' symbology to show a forwards, backwards or no arrow depending on one-way information attached to the street.

sDNA allows for elevation (z) data to be provided separately to grade separation. At first glance this seems odd, as these both measure the same thing. However, it is often the case that precise grade separation data is available, but only imprecise elevation data. In the case of a 3d network formed by draping a 2d network over a terrain model, the elevation differences of bridges and tunnels will not be captured, so grade separation is still needed to indicate these.

To display and check elevation data, it is best to view the network in 3d.

Definitive check for connectivity errors

The definitive way to check for connectivity errors is to get sDNA to calculate the connectivity of each link and see if it displays what you think the connectivity should be. Run sDNA Individual Line Measures to compute the connectivity of each link, and check it is as you expect. Link connectivity (LConn) is *the number of ends of other links that the link is attached to*.

Of course, checking connectivity by hand over an entire network may be too costly an operation for you to undertake. sDNA cluster tolerance, and external tools to *Break intersections* and fix *Topology* can fix connectivity errors automatically. Bridges and tunnels are usually few in number, so can be checked manually. With a reliable source of data (such as Ordnance Survey), we can usually trust that the source data is correct in any case.

A final way to check for connectivity problems in key locations, however, is to use sDNA to compute a simple model of angular betweenness. This takes some computation time, but the results should highlight all major routes through the network. If any of these look implausible – the routes you expect to be major are not, or vice versa – this may hint at a connectivity error on a major route.

Geometry errors at key locations

The final common error in spatial network analysis applies only to angular analysis, i.e., any analysis in which changes of direction are considered important. This includes any hybrid analysis that includes a component of angularity, but not Euclidean or topological analysis.

There are certain types of error in feature encoding that cause spurious changes of direction on simulated paths through the feature. Figure 5 gives two examples. In the first, what appears at one scale to be a crossroads is slightly staggered – so a route travelling from left to right has to negotiate two 90° turns when in reality, none are needed at all. This kind of problem is common even with usually reliable data sets, which may not have been designed with angular analysis in mind.

In the second example of Figure 5, a link is shown to have a zigzag feature almost obscured by an adjoining link. The zigzag may in fact completely overlap the other link, in which case it would be invisible to casual inspection (though overlaps should be fixed as a matter of course when fixing *Connectivity errors at key locations*).

One clue that almost always indicates a geometry error is a very short line. To find these, sort all polylines in your model by length, and inspect the shortest ones. Should they be there, or are they part of a staggered crossroads?

As with overlap errors, *Topology* tools help to fix geometry errors. All zigzags, staggered crossroads and short edges smaller than the tolerance will be removed. It is up to the user to select a suitable tolerance large enough to remove the errors you have observed in the data, but small enough to leave genuine features intact.

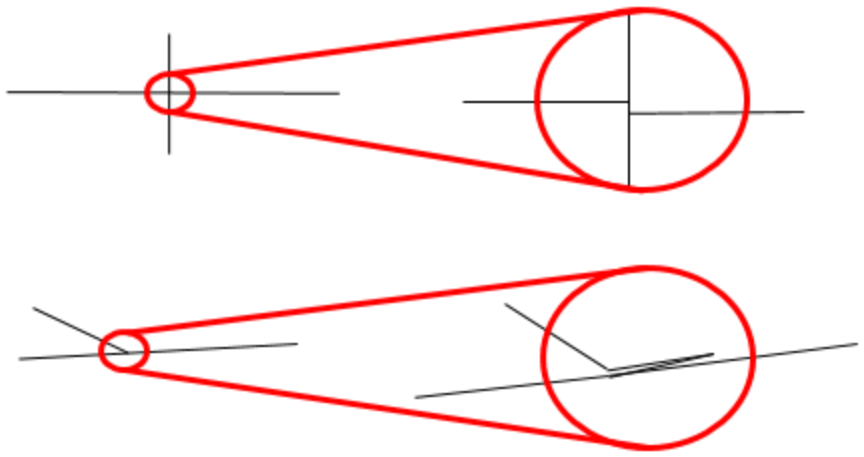

Fig. 5: Figure 8: Two kinds of geometry error. In the lower picture, the zigzag link may overlap the other link precisely, so be invisible to manual checking.

Traffic islands

A final topic to cover is that of traffic islands in road networks. Some data products (notably Ordnance Survey ITN) will encode larger traffic islands by splitting a link into two parallel parts for the length of the island ():num:*Figure #trafficisland*).

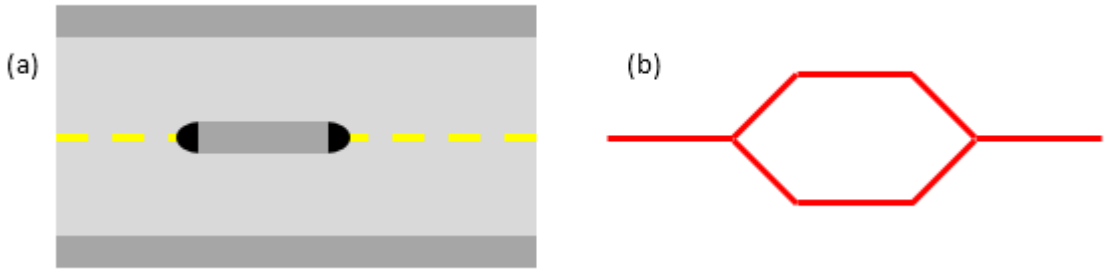

Fig. 6: Figure 9: Spatial network representation of traffic island. (a) road layout, (b) network representation

This encoding breaks angular analysis, by introducing a spurious 180° of turning in each direction along a road which is straight. The problem can be significant, as traffic islands are typically found on major routes through a city; the extra angularity introduced by traffic island encoding will cause sDNA to overestimate the metric for these roads and hence underestimate traffic flows.

sDNA Prepare includes a tool to fix traffic islands. You first need to compute a data field which is set to 1 for links which represent traffic islands, and 0 otherwise (this is usually easy to do from your source data). Then, run sDNA prepare to remove the islands.

1.2.2 Tools

The above discussion on network preparation has so far referred to tools for doing so only in the abstract. This section points to a few software packages that can achieve the desired aims.

Note that all of these tools can potentially corrupt grade separation data for bridges and tunnels.

Also note that fixing geometry problems using Topology tools will often fix connectivity problems at the same time.

Break intersections

Breaking intersections between lines means converting intersections from the type shown in Figure 1 to the type shown in Figure 2.

- **QGIS:** within the open source environment, using the `break` option on v.clean in the GRASS tools can be used to break polylines where they intersect. The GRASS tools are bundled with the free QGIS, though to display them it is necessary to switch the Processing toolbox to advanced mode. (This is necessary for *Using sDNA for the first time* in any case).
- **ArcGIS:** ArcGIS Advanced offers two tools to break intersections *en masse*. The simplest is to run ArcGIS Planarize with a tolerance of zero. The alternative is to use Arc Topology tools to detect and fix all intersections. We have found the latter is more flexible, but less able to deal with large numbers of intersections.
- **Autocad Map3d:** Autocad Map3d has a *Break lines at intersections* tool within the Drawing Cleanup toolkit.

Shared point line breaker

The `bpol` option on v.clean in the GRASS tools does exactly this. The GRASS tools are bundled with the free QGIS, though to display them it is necessary to switch the Processing toolbox to advanced mode. (This is necessary for *Using sDNA for the first time* in any case).

Topology

For the purpose of this document, we define Topology tools as those which can fix approximate (as well as actual) intersections and duplicates within a given cluster tolerance. These are very useful for cleaning up geometry errors such as those shown in Figure 4 and Figure 5. Note that sDNA *Prepare network* itself can fix intersections within a given tolerance *for link endpoints only*; for errors at other locations, use the following tools:

- **QGIS:** The `snap` option on v.clean in the GRASS tools will snap together portions of lines within a tolerance. This should be followed by `break` and `rmline`. The related tool `v.clean.advanced` will allow these operations to be combined.
- **ArcGIS:** Use ArcGIS Planarize or Arc Topology.
- **Autocad Map3d:** use *Apparent Intersections* in the Drawing Cleanup toolkit.

In all the above cases, take care to select a suitable tolerance or threshold value; large enough to fix errors but to leave genuine features intact.

1.2.3 Notes on specific formats

Address point data

To process address point data (e.g. floor area or population attached to building entrances), join the data of interest to the network using the Spatial Join function of your GIS. If greater than link level accuracy is required, split links into shorter segments.

OpenStreetMap (OSM)

Open Street Map (OSM) is currently the world's most prominent open mapping platform. A non-profit foundation registered in England and Wales since 2006, OSM passed the million-user mark in 2013, containing 21 million miles of road data and recruiting 1000 new contributors per day, resulting in ever increasing accuracy [OSM1]. More than 100 universities have research associated with Open Street Map [OSM2].

On the other hand, the crowd-sourced nature of OSM means that some unique problems are encountered in its usage. Data quality is not consistent, with more accuracy in some regions than others; there is also a lack of consistency between regions when it comes to the recording of link attributes. The definitive OSM source data is corrected and updated daily, although with a bias towards more updates in some areas than others.

These issues notwithstanding, we expect OSM to improve over time, and we have already found it to be very useful here at sDNA headquarters. In 2014 the author produced a model of the Cardiff city region based on OSM, that correctly predicted 90% of the variance in vehicle traffic flows, and 75% of the variance in flows of pedal cycles. In the UK, OSM is probably the most complete digital record of pedestrian and pedal cycle routes published to date.

Our experience with OSM has alerted us to the following pitfalls:

1. Spatial referencing in ArcGIS

   OpenStreetMap is stored and downloaded on the WGS84 datum. ArcGIS fails to correctly identify the transformation necessary to project OSM data to a national grid. The easiest way to resolve this is (before loading the data) to mark it as WGS84 using ArcCatalog. Once loaded, the data must be reprojected to a Euclidean (projected) coordinate system before analysing in sDNA.

   The free QGIS handles OSM data correctly, though it must still be reprojected before use with sDNA.

2. Connectivity and geometry errors

   As of November 2014, the OSM data for Cardiff contained a number of connectivity and geometry errors. These were fixed by planarizing in ArcGIS with a 1 metre cluster tolerance. It was first necessary to extract bridges and tunnels, to avoid planarizing these also.

See also the step by step guide, *Downloading and preparing data from OpenStreetMap*.

Update: as of 2020 many areas of OpenStreetMap are greatly improved and can be prepared correctly using the `bpol` option on v.clean in the GRASS tools.

Pedestrian networks

We have in the past produced a specification that deals with how to reliably create pedestrian networks for complex urban environments (joining interior and exterior networks in 3d).

1.3 Analysis: what the results mean

Before we start, you have prepared your network, haven't you? If you haven't, then go back to *Network Preparation*. It's very important.

This section provides an informal introduction to the outputs of sDNA *Integral Analysis*. Although it focuses or urban networks, the discussion is relevant to all domains of network analysis. A picture speaks a thousand words, so let's begin with one.

Fig. 7: Figure 1: Cardiff link density, 1km radius. Basemap (c) OpenStreetMap contributors

1.3.1 Links and network radius

The map of Cardiff coloured by link density at 1000m radius shown in Figure 7 makes a good starting point for a discussion about links and radius. This data was output by sDNA Integral with a label of *Links R1000*, or (in abbreviated format for shapefiles/QGIS) *Lnk1000*.

Firstly, we have the definition of network link. All links in a spatial network consist of a line joining a junction or dead end, to a junction or a dead end. Junctions and dead ends are the only things that links can join together: any line that doesn't join two of those things (or one of those things to itself) is not a complete link, though it will certainly be part of a longer line that does join two of those things, and which will therefore be a link.

Note also that links do not connect to junctions except at their endpoints. So in a road network, for example, what you might call a single street will in fact consist of several individual links. A street is defined by its name, a link by geometry alone.

We think link density is a good measure of urban density. The map above is based on a count, for each link, of how many links there are within a 1000m radius around it. Links are the fundamental unit from which spatial networks are built, and for that reason it makes sense to standardize on them as a unit of description. Also, link density tends to hold information about the network. For example, in a road network, link density can correlate over 90% with the density of houses and jobs[1]. Thus link density, in the absence of any other information, gives a pretty good indication of the amount of human activity surrounding any particular point.

The other important thing to discuss in Figure 7 is what the R1000 – the 1000m radius – means. A typical assumption in network analysis is that entities can only travel through the network: cars can't drive off roads, fish can't swim through rock, etc. So the 1000m radius is different for each link, and means "all points that can be reached by travelling 1000m *through the network* from that link".

For the uninitiated, a network radius is quite hard to visualise. So let's be clear about this – it's definitely not a circle.

[1] Chiaradia, Alain J., Hillier, Bill, Schwander, Christian and Barnes, Yolande 2013. Compositional and urban form effects on residential property value patterns in Greater London. Proceedings of the ICE - Urban Design and Planning 166 (3) , pp. 176-199. 10.1680/udap.10.00030

It's actually a set of link cut points on the network at a variable distance, as the crow flies, from the origin. But some find it easier to visualise a sort of wobbly blob surrounding the origin. Figure 8 illustrates this.

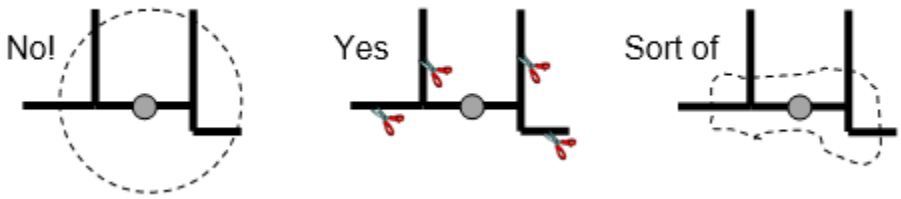

Fig. 8: Figure 2: Illustration of what a network radius is, and isn't.

Lots of output measures from sDNA contain "R1000" - or R followed by a number - as a suffix (in shapefiles/QGIS the R is dropped to save space). These are all radial measures – referring to a specific radial distance (usually expressed in network Euclidean terms, though alternatives are possible). Sometimes output measures contain "Rn". Radius n means everything: the entire network reachable from the link in question, excluding any parts that aren't connected.

In continuous space analysis, the radius has a "c" put after it - for example, R500c, R1000c, or Rnc (which as it happens is equivalent to Rn). This shows that the radius is computed using Continuous Space (for more details see *Radius*). Normally sDNA analysis treats each link as a discrete entity – it is either inside the radius, or it isn't. Continuous space analysis treats links as continuous entities; if part of the link falls inside the radius, and part outside, then the link is split on-the-fly and we count only the part of that falls inside. Continuous space should always give the most accurate results, but discrete space is faster to compute.

Besides link density, sDNA produces numerous other measures that quantify the network in the radius:

- *Total Length* is self explanatory
- *Number of junctions* is self explanatory
- *Total Weight* is the total weight customizable by the user
- *Connectivity* is the total number of link ends connected at junctions

1.3.2 Betweenness

The flow model inherent in sDNA is based on Betweenness. Figure 9 shows an example. Note this is link-weighted, so the result you see is *based on the shape of the network alone*.

Betweenness analysis assumes your network is populated with entities that go from everywhere to everywhere else, subject to a maximum trip distance determined by the radius. We assume these entities travel via the shortest possible path – and we call such a path a geodesic. But how we define "shortest" may vary. We call the definition of distance a *metric*; currently sDNA supports

- *Euclidean metrics*, taking the shortest physical distance possible
- *angular metrics*, which minimize the amount of turning both on links and at junctions
- *custom* metrics based on user data
- other specialist metrics

As a first approach to most urban network problems, we really like using angular metrics. Pedestrians, unless they know an area very well, will tend to follow the shortest angular paths, because they are easier to remember ("second on the right then straight on 'till morning" – Peter Pan would probably have got lost had he tried to take a short cut requiring more complex directions). Drivers of vehicles also tend to follow angular geodesics, but for a different reason – straight roads through a city tend to be faster, on average. So we have set the default metric in sDNA to be angular.

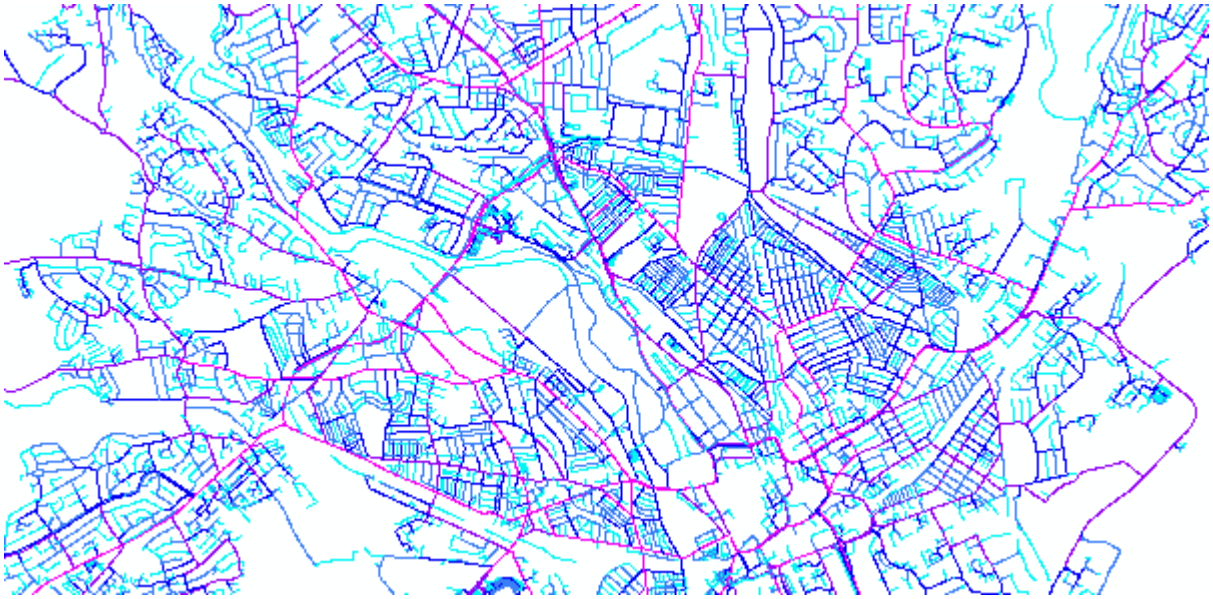

Fig. 9: Figure 3: Cardiff Angular Betweenness, Radius 10km. Basemap (c) OpenStreetMap contributors

Take care, however, to set the radius for your intended purpose! A realistic indicator of pedestrian flow might be Betweenness Ang R800, while a realistic indicator of vehicle flow might be Betweenness Ang R2000, or higher. But fish, for example, probably don't care much about going around corners in rivers, so Euclidean analysis might be more appropriate. Pedestrians commuting to and from work tend to follow Euclidean geodesics too, because they know all the short cuts.

Note that in betweenness analysis, we are usually looking at an intermediate link in a trip, not the origin or destination. This means that the radius has a subtly different meaning: a calculation of betweenness R500 does not involve all possible trips within 500m of the link you are observing, but involves all possible trips that pass through the link you are measuring with a maximum length of 500m (almost: see *Geodesic* for more details).

A final question in betweenness analysis is what weight to assign to each geodesic. The simplest answer is to assign a weight of 1 to every pair of links. As the density of jobs and homes is strongly related to the density of network links, betweenness weighted in this manner usually correlates well with traffic flows. But what if network length is important to you? Let's say you want to assume longer streets act as origins and destinations for more people, or you're analyzing rivers, etc. Or maybe you have custom weights - census data for example, or building entrances. In this case we assume the number of entities making the trip is proportional both to the weight of the origin and the weight of the destination, so we multiply the origin weight and destination weight together to compute the betweenness weight.

Interestingly though, such flows scale with the square of the number of links in the network, implying that more dense areas contain more activity *per link* - in other words, an opportunity model. (To a physicist, there is a problem with the units: Betweenness is measures in units of weight squared, not weight). If this is not desirable then instead use a Two Phase Betweenness model. These assume each origin has a fixed amount of weight it can add to betweenness – in the case of cities, a fixed quantity of journeys that the people living there will realistically make. This fixed quantity is then divided proportionately among the destination weights accessible from that origin for the current radius; this can be described as a generation-distribution model.

Complementing Two Phase Betweenness is Two Phase Destination – the quantity of visits each destination gets under the two phase model. In the normal betweenness model, each destination will be visited by everything within radius x, so this measure would be equivalent to *Weight*. But in a two phase model, destinations must compete with other destinations for the attention of origins, and TPDestination measures the total flow to the destination taking account of this competition. A real world example would be the case of high street shops - with small destination weight - trying to compete with an out-of-town retail park - with large destination weight. Both may have lots of population in their catchment (which will show up in the Links Rx measure) but the town centre loses out to the new development (which

shows up in the TPDestination Rx measure).

That said, all the forms of betweenness tend to correlate quite well with actual flows of pedestrians, vehicles etc.

1.3.3 Closeness

Closeness, like betweenness, is a form of network centrality. It matches commonly held notions of accessibility.

Mean Distance is perhaps one of the most common closeness measures in the literature. It measures the difficulty, on average, of navigating to all possible destinations in radius x from each link. Technically then it's a form of farness not closeness: this only means that big numbers mean "far" instead of "close".

"Difficulty" is defined in terms of the same metric used for Betweenness: Euclidean, angular, custom, hybrid, and so on. sDNA names its closeness outputs Mean Angular Distance (MAD), Mean Euclidean Distance (MED), Mean Custom Distance (MCD), etc.

Note that the "mean" of mean distance is weighted by the weight of destinations.

1.3.4 Other measures

sDNA produces numerous other output measures, for a full description refer to *Analysis: full specification*.

sDNA can also display the geometries of geodesics, network radii and convex hulls used in the analysis, as well as producing accessibility maps for specific origins and destinations.

1.3.5 Summary of measures

Measure	Abbreviations	Description
Closeness (mean distance)/Farness	MAD, MED, MCD, MHD	Inverse of closeness, accessibility
Mean Geodesic Length	MGLA, MGLE, MGLC, MGLH	As closeness but always output in units of length even for non-Euclidean geodesics
Network Quantity Penalized by Distance	NQPDA, NQPDE, NQPDC, NQPDH	Sum of network quantity over distance
Betweenness	BtA, BtE, BtC, BtH	Flow model, through-movement
Two Phase Betweenness	TPBtA, TPBtE, TPBtC, TPBtH	Generation-distribution flow model
Two Phase Destination	TPDA, TPDE, TPDC, TPDH	Floating catchment, competitive accessibility
Mean Crow Flight	MCF	Mean distance to destinations as crow flies
Diversion Ratio	DivA, DivE, DivC, DivH	Mean ratio of geodesic length to crow flight distance
Convex Hull Area	HullA	Area of convex hull of radius
Convex Hull Perimeter	HullP	Perimiter of convex hull of radius
Convex Hull Max Radius	HullR	Furthest extent of convex hull; crow flight distance for most efficient route
Convex Hull Bearing	HullB	Bearing of furthest extent / most efficient route
Convex Hull Shape Index	HullSI	Describes shape of hull; ranges from 1 for circle to infinity for straight line
Links	Lnk	Links in radius
Length	Len	Length in radius
Weight	Wt	Weight in radius
Angular Distance	AngD	Angularity in radius
Junctions	Jnc	Junctions in radius
Connectivity	Conn	Connectivity in radius
Line Length	LLen	Length of individual polyline
Line Connectivity	LConn	Connectivity of individual polyline
Line Angular Curvature	LAC	Angularity of individual polyline
Line Hybrid Metric	HMf, HMb	Hybrid metric of individual polyline
Line Sinuosity	LSin	Diversion ratio for individual polyline
Line Bearing	LBear	Bearing of individual polyline
Link Fraction	LFrac	Fraction of link represented by polyline

1.3.6 Calibration

sDNA produces network statistics that correlate to real phenomena such as traffic flows or land prices, but if you are interested in predicting these phenomena, you will need to know how to convert sDNA outputs into a prediction. This is done by regression analysis, a huge topic with which some users of sDNA will already be familiar, but others won't.

sDNA provides two tools to help with this process, *Learn* and *Predict*. *Learn* takes a sample of real data and creates a model linking it with the output of sDNA. *Predict* takes a model created by *Learn* and applies it to new areas where data is not available, based on an sDNA analysis of those areas.

Of course, it is also possible to use any third party regresion software of your choice to perform these tasks.

1.4 Analysis: full specification

This section outlines a more formal specification of the network analysis concepts on which sDNA Integral, Accessibility Map, Geodesics, Network Radii and Convex Hulls are built.

Note that as of sDNA Version 3, some of this specification has changed, in particular the handling of split links (the rationale for which is explained in a blog post). The specification for versions 1 and 2 is still available.

1.4.1 Definitions

Link

One or more polylines spanning the gap between two junctions, or a junction and an end point.

sDNA accepts data in *link-node format* (though strictly speaking, the nodes aren't necessary). It also accepts data in *polyline-node* format. Note that we make a distinction between *link* and *polyline*: some users of these formats don't, and will treat them as the same thing. This is importatant for your understanding of sDNA, but not important when using the software as it will accept either format.

To represent transport networks, polylines can be centred on a road or path centre line. Polylines can also be used to outline pedestrian paths and pedestrian crossings, thus creating a pedestrian centre line network. The largest to date, to our knowledge, is 1400km long; created for the Commnunauté d'Agglomération de Saint Quentin en Yvelines, Paris, France.

Polyline

A link consists of one or more *polylines*. This matches the GIS or CAD notion of a polyline: a continuous chain of line segments treated as a single unit, usually with a number of attached data fields. Polylines vary in length, in angularity (directness/sinuosity), in connectivity (number of other ends of polylines connected).

Internally to sDNA, polylines are treated as indivisible origins; all flows originating from a polyline are treated as originating from its middle (which is defined as the Euclidean or angular centre according to the Distance Metric). As destinations, polylines are treated as indivisible in Discrete Space mode, or divisible in Continuous Space mode.

Junction

A point where at least 3 link ends are connected.

Distance Metric

A metric is a function used to define what it means for points on the network to be "close to" or "far away from" each other. sDNA currently supports

- ***Euclidean metric*** Distance measured along the network in the distance units the network is defined in. This is the everyday notion of distance
- ***Angular metric*** Distance measured in terms of angular change; i.e. corners on links and turns at junctions
- ***Custom metric*** Distance measured according to a user chosen data field defined on Polylines
- ***Other metrics*** We provide some other specific metrics suitable for urban network analysis

Radius

Analyses are performed from each origin within a user defined radius expressed in the spatial units of the data. Usually the set of origins is equal to the set of Polylines, so an analysis is conducted around each Polyline.

The user will usually wish to express a radius in metres, therefore care must be taken to project spatial data into a coordinate system measured in metres before using sDNA. Radius can be understood as variable floating catchment area, a locality of analysis, or a maximum trip distance.

Two modes exist for handling of the radius boundary:

Discrete space all polylines are included in the radius of each origin if and only if their centres fall within the radius

Continuous space polylines are dynamically split, and only those parts that exactly fall within the radius of each origin are included in the analysis of that origin. Fractional polylines have weights adjusted downwards according to the fraction of the link length they represent.

Continuous space takes slightly more computation than discrete space, but is recommended for analyses in which the radius has the same (or smaller) order of magnitude as the longest polylines. This usually implies small scale analysis e.g. a pedestrian or cyclist network.

A radius is expressed as a Euclidean distance by default, in units of the source data projection. If *banded radius* is used, each radius excludes links from the next smallest radius.

Geodesic

A geodesic is the shortest route between two points on the network, according to a given Distance Metric.

Various special cases exist where this is not strictly true. Note that when using non-Euclidean metrics, the network radius may be shorter than a geodesic. As the network radius is supposed to specify a locality of analysis, we constrain such geodesics to use only network within the radius. This behaviour can be modified in *Advanced configuration and command line options*. The implications of handling these edge cases are explored in[1].

1.4.2 Weighting

The importance of each origin and destination in the analysis is determined by weighting. Weights can be

- per polyline
- per link
- per unit length

All of these can be based on custom data. If weights are not given, all weights will default to 1, though this default can be interpreted (as above) per polyline, link or unit length.

The choice of weighting should reflect what you consider to be the best measure of *network quantity* in your analysis: number of links, length, or a custom defined property (such as population plus number of jobs, or number of address points). The recommended choice for urban networks – if you don't have actual census data – is number of links. This is because link density increases with number of jobs and population: thus measuring network by the number of links goes some way towards capturing these other variables through network geometry.

[1] Cooper (2015) Spatial localization of closeness and betweenness measures: a self-contradictory but useful form of network analysis. International Journal of Geographical Information Science 29 (8). http://www.tandfonline.com/doi/abs/10.1080/13658816.2015.1018834

Weighting	Output suffix	Origin/Destination weight fields	Result
Link		Left blank	All origin and destination links have weight of 1. Recommended default for analysis without weight data
Length	Wl	Left blank	All origin and destination links have weight equal to their length. Alternative default for instances where length better represents urban density than link count
Polyline	Wp	Left blank	All polylines receive weight of 1 (not recommended, we don't know of a logical use for this)
Link		Specify user weight data	All origin and destination links have weight equal to the custom weight provided by user defined field (or averaged for links represented by multiple polylines). Recommended for attaching weights based on land use data
Length	Wl	Specify user weight data	All origin and destination links have weight equal to the custom weight provided by user defined field, multiplied by their length. In other words, the user has specified weight per unit length. Alternative for attaching weights based on land use data
Polyline	Wp	Specify user weight data	All polylines receive weight equal to the custom weight. Recommended for attaching address point or census data at sub link level

Note that address points (attached to individual buildings) can be processed by using a GIS Spatial Join to transfer data such as floor area from the point data to the network. If sub link accuracy is needed, links should be split into multiple sub links.

1.4.3 Handling of one-way systems

When computing geodesics, the analysis respects specified one-way and vertical one-way links. However, when computing the contribution of a single link to its own closeness/betweenness/weight in radius etc, it is assumed that all points on the link are directly reachable from one another regardless of one-way status. This is to maintain consistency with origin approximations and choice of link centres, and the handling of other micromodelling situations within sDNA. In the absence of black/white holes (points within a one way system that are impossible to escape/enter), this will result in a correct computation of links/length/weight in radius for all sufficiently large radii (that is to say, if the radius exceeds the maximum geodesic from any link to itself respecting the one way system). Origin self-closeness/self-betweenness usually makes only a small contribution to the overall analysis and is included principally for consistency.

1.4.4 Mathematical definition of outputs

Notation

In the following sections,

- The set of polylines in the global spatial system is denoted N
- The set of polylines in the network radius from link x is denoted R_x
- The proportion of any polyline y within the radius is denoted P(y). In discrete space analysis, this always equals 0 or 1, i.e. $y \in R_x \Leftrightarrow P(y) = 1$. In continuous space, $0 \leq P(y) \leq 1$
- Length of a polyline y is denoted $L(y)$

- The distance according to a metric M, along a geodesic defined by M, between an origin polyline x and a destination polyline y is denoted $d_M(x, y)$
- The network Euclidean distance along a geodesic defined by a metric M, between an origin polyline x and a destination polyline y is denoted $d_M^E(x, y)$
- Weight of a polyline y is denoted $W(y)$.

$W(y)$ is computed differently according to the weighting scheme. If

$$U(y) = \begin{cases} \text{the user defined weight for polyline } y, \\ \text{if defined} \\ 1, \\ \text{otherwise} \end{cases}$$

then

$$W(y) = \begin{cases} U(y), \\ \text{for polyline weighting} \\ U(y)L(y), \\ \text{for length weighting} \\ U(y)\frac{L(y)}{L(Z)}, \\ \text{for link weighting} \end{cases}$$

where $L(Z)$ is the length of the *link* Z formed of one or more polylines P such that $y \in P$

Centrality measures

Farness

Farness is measured as the Mean Angular, Euclidean, Custom or Hybrid distance according to the chosen distance metric. It is abbreviated as MAD, MED, MCD and MHD respectively.

$$\mathbf{Farness}(x) = \frac{\sum\limits_{y \in R_x} d_M(x, y)W(y)P(y)}{\sum\limits_{y \in R_x} W(y)P(y)}$$

Note that on average, the distance of traversing between two arbitrary points within the same link is $1/3$ the distance of traversing the entire link. The contribution of the origin link to its own farness is included in this manner.

Closeness

sDNA doesn't measure closeness, it measures farness, which tells you exactly the same thing in a different way. The literature often defines closeness as $1/\mathbf{farness}$, though this has an exponential distribution so statistically is harder to work with. An alternative definition of closeness that doesn't suffer from this problem is $-\mathbf{farness}$. We don't tend to use either, preferring to use either Farness or NQPD for the same purpose.

Mean geodesic length

Mean geodesic length (MeanGeoLen or MGL) is the mean length (always in Euclidean metric) of all geodesics in the radius (defined by the chosen metric).

$$\mathbf{MGL}(x) = \frac{\sum\limits_{y \in R_x} d_M^E(x,y)W(y)P(y)}{\sum\limits_{y \in R_x} W(y)P(y)}$$

MGL can be used as an invariant measure to compare geodesics from different hybrid metrics. For example, a cyclist metric that accounts for road traffic can be compared to a cyclist metric without motor vehicle traffic by calculating $MGL_{\text{traffic}}/MGL_{\text{no traffic}}$. This would show areas where cyclists must make large detours to avoid motor vehicle traffic.

Network quantity penalized by distance (gravity model)

NQPD is a form of closeness, commonly referred to as a gravity model, that takes into account both quantity and accessibility of network weight. By contrast, Farness takes into account only accessibility, while Weight takes into account only weight.

$$\mathbf{NQPD}(x) = \sum_{y \in R_x} \frac{(W(y)P(y))^{\text{nqpdn}}}{d_M(x,y)^{\text{nqpdd}}}$$

Note that on average, the distance of traversing between two arbitrary points within the same link is $1/3$ the distance of traversing the entire link. The contribution of the origin link to its own NQPD is included.

nqpdn and nqpdd default to 1, but can be set to other values in advanced config (they stand for NQPD numerator and denominator, respectively). The problem, for any given application, is determining the correct values to use for each i.e. the relative importance of network quantity and accessibility. To answer that question we recommend approximating NQPD with a multivariate translog linear regression based on Farness and Weight, i.e. using the model

$$\log(\text{variable of interest}) = \beta_0 + \beta_1 \log(\mathbf{farness}) + \beta_2 \log(\mathbf{weight}) + \epsilon$$

Once suitable values for β have been obtained, these can be applied to nqpdn $\approx \beta_2$ and nqpdd $\approx -\beta_1$.

Betweenness

Betweenness counts the number of geodesic paths that pass through a vertex, i.e, the number of times the vertex lies on the shortest path between other pairs of vertices.

$$\mathbf{Betweenness}(x) = \sum_{y \in N} \sum_{z \in R_y} W(y)W(z)P(z)OD(y,z,x)$$

where

$$OD(y,z,x) = \begin{cases} 1, \\ \text{if x is on the first geodesic found from y to z} \\ 1/2, \\ \text{if } x = y \neq z \\ 1/2, \\ \text{if } x = z \neq y \\ 1/3, \\ \text{if } x = y = z \\ 0, \\ \text{otherwise} \end{cases}$$

Note that the geodesic endpoints are y and z, *not* x where the betweenness is being measured. The contributions of $1/2$ to $OD(y,z,x)$ reflect the end links of geodesics which are traversed half as often on average, as journeys begin and end in the link centre on average. The contributions of $1/3$ represent origin self-betweenness.

Note that, in cases where a number of equal length geodesics exist between an origin and destination pair, sDNA will only consider the first such geodesic found. This differs from some literature where betweenness is distributed over all geodesics of equal length. If this is of concern (e.g. analysing a perfect grid pattern) we recommend adding a small amount of randomness to the analysis using a hybrid metric, and if necessary, combining this approach with oversampling. (Oversampling is not usually necessary as randomness is reapplied per origin; i.e. every link analyzed represents a different sample of the random distribution even with no oversampling). Randomness and oversampling can be set in *Advanced configuration and command line options*.

Two phase betweenness

Two phase betweenness (TPBt) represents Betweenness, but rather than being weighted by a product of origin and destination weights, the origin weight is distributed over destination weights. Since implementing this I have realized that it represents the flows from what is referred to in some older literature as the Huff model for accessibility. It is thus the sum of geodesics that pass through a link x, weighted by the proportion of network quantity accessible from geodesic origin y that is represented by geodesic destination z.

$$\mathbf{TPBt}(x) = \sum_{y \in N} \sum_{z \in R_y} OD(y,z,x) \frac{W(z)P(z)}{\text{total weight}(y)}$$

where total weight(y) is the total weight in radius from each y.

Two phase destination

Two phase destination (TPD) measures the proportion of origin weight received by each destination in the two phase betweenness model. It is similar to a two-step floating catchment, and the Huff accessibility model.

$$\mathbf{TPD}(x) = \sum_{y \in N} \sum_{x \in R_y} \frac{W(x)P(x)}{\text{total weight}(y)}$$

In a normal betweenness analysis, this quantity would be equivalent to Weight, but as geodesic weight from each origin is limited in the two phase model, the weight transferred to destinations along geodesics becomes dependent not only on the weight within radius of the destination, but also on what that destination is competing with. Thus this measure is more discriminating of spatial hierarchy than the Links, Length and Weight measures described below.

Note that TPBt has units of – and scales with - network quantity, rather than the square of network quantity as is the case with standard Betweenness. Thus it corresponds to transport models with trip generation and distribution phases, while normal betweenness can be seen as an opportunity model.

Network detour analysis

Network detour analysis compares straight line distance to actual network distance, answering the question, “By how much does the network deviate from the most direct path?”

Mean Crow Flight

Mean Crow Flight (MCF) is the mean of the crow flight distance between each origin and all links within the radius.

$$\mathbf{MCF}(x) = \frac{\sum_{y \in R_x} CFD(x,y)W(y)P(y)}{\sum_{y \in R_x} W(y)P(y)}$$

where $CFD(x,y)$ is the crow flight distance between the centers of x and y.

MCF can be compared to Mean Geodesic Length (MGL); that is to say, $\frac{MGL}{MCF}$ gives a measure of the extent to which geodesics must divert from desire lines.

Diversion Ratio

Diversion Ratio (Div) is the mean ratio of geodesic length to crow flight distance over all links in the radius. It differs from $\frac{MGL}{MCF}$ in that ratios are computed individually before averaging.

$$\mathbf{Div}(x) = \frac{\sum\limits_{y \in R_x} \frac{d_M(x,y)}{CFD(x,y)} W(y)P(y)}{\sum\limits_{y \in R_x} W(y)P(y)}$$

Network shape analysis

Network shape analysis refers to the form of the overall spatial footprint of the network within the radius. This can be used to compute measures of efficiency, for example, $\mathbf{Convex\ Hull\ Area/Length}$ provides a measure of the coverage of Eucidean space per unit length of network.

All of the following measures are based on a 2-d convex hull of all points within the network radius. For 3-d networks, the network is projected onto the x-y plane before computing a convex hull.

Convex Hull Area

Convex Hull Area (HullA) is the area of the convex hull covered by the network within the radius.

Convex Hull Perimeter

Convex Hull Perimeter (HullP) is the perimeter of the convex hull covered by the network within the radius.

Convex Hull Maximum Radius

Convex Hull Maximum Radius (HullR) is the distance (as the crow flies) from the origin to the point where the convex hull has its greatest radius (as the crow flies). In other words, it is the largest crow flight distance to any point within the network radius, and as such represents the single route accessible from the origin that can cover the most distance as the crow flies.

Convex Hull Bearing

This is the compass bearing of the *Convex Hull Maximum Radius*, as measured from the positive y direction of the projected grid (this is usually grid north).

Convex Hull Shape Index

This is defined as

$$\textbf{Hull Shape Index} = \frac{(\textbf{Hull Perimeter})^2}{4\pi(\textbf{Hull area})}$$

Note that the minimum possible shape index is 1 (for a circle), and the maximum is infinity (for a straight line).

Radius description measures

For each radius from each origin the following measures are given:

Links

Links (Lnk) is the number of links in the radius, $\sum_{y \in R_x} P(y)$

Length

Length (Len) is the total network length in the radius, $\sum_{y \in R_x} L(y)P(y)$

Weight

Weight (Wt) is the total weight in the radius, $\sum_{y \in R_x} W(y)P(y)$. If you wish to normalize any other output measure for quantity of network, this is the best control to use as it adapts to the analysis type as appropriate.

Angular Distance

Angular distance (Ang Dist or AngD) is the total angular curvature on all links in the radius $\sum_{y \in R_x} d_\theta(y_R)$, where y_R is the proportion of y that falls within the radius only.

Junctions

Junctions (Jnc) counts the number of junctions in the radius. Note that only junctions between links, not polylines, are counted.

Connectivity

Connectivity (Con) is the total connectivity in the radius: sum of number of links ends connected at each junction. Note that one way streets count as half a link end in this measure (unlike LConn where they are counted fully).

Individual polyline descriptive measures

Line Length

Line length (LLen) is the Euclidean length of polyline, $L(y)$.

Line Connectivity

Line Connectivity (LConn) is the number of other line ends to which this line is connected. Also called *degree centrality*.

Line Angular Curvature

Line angular curvature (LAC) is the cumulative angular curvature along the full length of the line, in degrees: $d_\theta(y)$.

Line Hybrid Metrics

In a hybrid analysis, the hybrid metrics for the polyline are given: Hybrid Metric forward (HMf) and Hybrid metric backward (HMb). The metric can be different per direction due to height gain, or custom behaviour that relates to direction of traversal, for example escalators, traffic priority or one way tolls.

Line Sinuosity

Line Sinuosity (LSin) is the line length divided by distance as the crow flies between its endpoints. Similar to diversion ratio but for a single line only.

Line Bearing

Line Bearing (LBear) is the compass bearing between line endpoints, as measured from the positive y direction of the projected grid (this is usually grid north).

Link Fraction

Link Fraction (LFrac) is the proportion of a link that this line represents, $P(y)$.

1.4.5 List of outputs and abbreviations

Abbreviation	Description
AngD	Angular Distance in Radius
BtA	Betweenness Angular
BtC	Betweenness Custom
BtE	Betweenness Euclidean
BtH	Betweenness Hybrid
Conn	Connectivity in Radius
DivA	Diversion Ratio in Radius Angular
DivC	Diversion Ratio in Radius Custom
DivE	Diversion Ratio in Radius Euclidean
DivH	Diversion Ratio in Radius Hybrid
HMb	Line Hybrid Metric (backwards direction)
HMf	Line Hybrid Metric (forwards direction)
HullA	Convex Hull Area
HullB	Convex Hull Bearing of Maximum Radius

Continued on next page

Table 1 – continued from previous page

Abbreviation	Description
HullP	Convex Hull Perimeter
HullR	Convex Hull Maximum (Crow Flight) Radius
HullSI	Convex Hull Shape Index
Jnc	Junctions in Radius
LAC	Line Angular Curvature
LBear	Line Bearing
LConn	Line Connectivity
Len	Length in Radius
Lfrac	Link fraction (for current line)
LLen	Line Length
Lnk	Links in Radius
LSin	Line Sinuosity
MAD	Mean Angular Distance in Radius
MCD	Mean Custom Distance in Radius
MCF	Mean Crow Flight Distance in Radius
MED	Mean Euclidean Distance in Radius
MGLA	Mean Geodesic Length in Radius Angular
MGLC	Mean Geodesic Length in Radius Custom
MGLE	Mean Geodesic Length in Radius Euclidean
MGLH	Mean Geodesic Length in Radius Hybrid
MHD	Mean Hybrid Distance in Radius
NQPDA	Network Quantity Penalized by Distance in Radius Angular
NQPDC	Network Quantity Penalized by Distance in Radius Custom
NQPDE	Network Quantity Penalized by Distance in Radius Euclidean
NQPDH	Network Quantity Penalized by Distance in Radius Hybrid
SAD	Sum of Angular Distance in Radius *
SCD	Sum of Custom Distance in Radius *
SCF	Sum of Crow Flight Distance in Radius *
SED	Sum of Euclidean Distance in Radius *
SGLA	Sum of Geodesic Length in Radius Angular *
SGLC	Sum of Geodesic Length in Radius Custom *
SGLE	Sum of Geodesic Length in Radius Euclidean *
SGLH	Sum of Geodesic Length in Radius Hybrid *
SHD	Sum of Hybrid Distance in Radius *
TPBtA	Two Phase Betweenness Angular
TPBtC	Two Phase Betweenness Custom
TPBtE	Two Phase Betweenness Euclidean
TPBtH	Two Phase Betweenness Hybrid
TPDA	Two Phase Destination Angular
TPDC	Two Phase Destination Custom
TPDE	Two Phase Destination Euclidean
TPDH	Two Phase Destination Hybrid
Wl	Weighted by Length (as opposed to Link)
Wp	Weighted by Polyline (as opposed to Link)
Wt	Weight in Radius

*Outputs marked * are only shown if* `outputsums` *keyword is given in Advanced configuration and command line options.*

1.5 Troubleshooting Models

1.5.1 Is the model wrong, or am I?

If a model doesn't meet expectations then it is possible that either expectations or the model are out of line with reality. The difficulty is, of course, how to tell which. In cases where predictions can be compared to actual measurements, these can be used to determine what is wrong with the model. Are all outliers in a cycle model on a certain type of road, for example? The model is probably over or underestimating the effect of traffic. All in a certain part of town? The model could be misestimating the weight of a particular origin or destination. Isolated errors? These can be caused by the spatial network model misrepresenting what is actually there; for example, roads disconnected in the model when they are connected in reality.

In other cases, we have no predictions with which to be sure, but the model doesn't seem to match expectations. Two techniques can be used to drill into why the model predicts what it does; and hence help decide whether the results are likely to be correct.

Links with low flow in the model which we didn't expect

If a flow is unexpectedly low, then you as the user likely have an expectation of which origins and destinations should cause that flow. sDNA allows plotting of individual routes between origins and destinations, so it is possible to select some and try it.

1. Pick an origin/destination pair. Note down their object IDs in the network dataset. Suppose for this example, the origin has id 482 and the destination is 9907.
2. Run sDNA Geodesics *with the same configuration as the model run*. :
    a. Add "n" to the list of radii, e.g. 1000, 2000, 3000, 4000, n
    b. Set origins to 482 and destinations to 9907 (or whichever origin and destination id you noted down in step 1)

Be careful not to run Geodesics without specifying an origin and destination – or a LOT of data will be generated (routes for every origin/destination pair).

On inspecting the geodesic, one of two errors may be apparent:

a. Geodesic is generated for radius n but not the radius used in analysis. This means the endpoints were too distant for the chosen radius.

b. Geodesic is generated between the endpoints but not passing through the expected link. This implies that the analysis metric for routes through the expected link must be higher than the analysis metric for the actual geodesic. Inspecting the geodesic's analysis metric and compare to metrics generated from the expected route. The latter is shown for each link in the original cycle model output, as *Hybrid Metric Fwd (HmF)* and *Hybrid Metric Bwd (HmB)*. Inspection may reveal why the model regards the unexpected route as preferable, or may hint at problems in the spatial model.

Links with high flow in the model which we didn't expect

If a flow is unexpectedly high, run a betweenness or geodesics analysis with an *intermediate link filter* to find out where the flow comes from.

1. Create a new data field on the network, set to 0 everywhere but 1 on the link we are expecting.
2. Run sDNA Integral or Geodesics with an intermediate link filter specifying the new data field.

sDNA Integral will produce a map of flows passing through the link; sDNA Geodesics will allow identification of individual paths. Either way, trace the source of the predicted flows to establish whether or not there is an error; and if so whether it is in the origin/destination weighting, the analysis metric affecting the choice of routes, or the network geometry.

1.6 Installation and first usage

1.6.1 Requirements

sDNA requires Windows XP or later. We suggest a minimum of 1GB RAM, but obviously faster computers and more memory will be of use in analyzing larger networks. sDNA can run in 32- or 64-bit mode depending on the host application.

The toolbox can be used in any of a number of ways:

- As a plug-in to ArcGIS 10.1 or later, or
- As a plug-in to ArcGIS Professional, or
- As a plug-in to QGIS 2.0 or later, or
- As a plug-in to Autocad (various versions; 2010-2013 have been tested), or
- From the windows command line, or
- From sdnapy, the python API to sDNA.

Note that Autocad usage is limited to network shape analysis using sDNA Prepare and Integral. For more detailed analysis, we recommend using Autocad Map3d or other suitable software to convert CAD to 3d Shapefile, then processing in the free QGIS.

1.6.2 Installation

sDNA Open can be downloaded an installed from Github. You will need a serial number; these can be obtained for free.

Further steps are then needed depending on how you plan to use sDNA.

1.6.3 Using sDNA for the first time

How you use sDNA depends on your host application.

ArcGIS 10.x

1. From inside ArcGIS, go to ArcToolbox.
2. Right click the root of the ArcToolbox tree and choose `Add Toolbox...`".
3. Navigate to the place you have installed sDNA (usually `c:\Program Files (x86)\sDNA`) and select the toolbox `sdna.pyt`.
4. (Optional) Repeat steps 2 and 3 to add the toolbox `sDNA_ArcGIS_extra_tools.tbx`.
5. (Optional) To permanently add sDNA to ArcToolbox, right click the root of ArcToolbox and choose `Save settings` → `to Default`.

sDNA appears as a set of tools within ArcToolbox. Results can be displayed using the `Symbology` tab of the `Layer Properties` dialog. If you are not familiar with using tools from ArcToolbox, or changing layer symbology, visit the *ArcGIS Desktop Help* website for further details.

ArcGIS Professional

In ArcGIS Professional, external toolboxes appear in the Catalog rather than with ESRI's own geoprocessing tools.

1. From inside ArcGIS, navigate to the "View" ribbon and choose "Catalog Pane" to open the catalog.
2. In the Catalog pane, right click on "Toolboxes", Choose "Add Toolbox"
3. Navigate to the place you have installed sDNA (usually `c:\Program Files (x86)\sDNA`) and select the toolbox `sdna.pyt`.

The tools can then be used from the catalog.

QGIS

1. From inside QGIS, choose `Plugins` → `Manage and install plugins...`. At present, QGIS support is considered experimental, so go to `Settings` and click `Also show experimental plugins`.
2. Type "sdna" into the search box; you should find the Spatial Design Network Analysis plugin. Different sDNA plugins should appear depending on which version of QGIS you are using (2 or 3).
3. Click `Install Plugin`, then `Close`
4. Go to `Processing` → `Toolbox` to show the processing toolbox
5. At the bottom of the processing toolbox, change from `Simplified interface` to `Advanced interface`
6. "Spatial Design Network Analysis" should now appear in the processing toolbox
7. Go to `Processing` → `Options` → `General` and ensure `Keep dialog open after running an algorithm` is switched on.

Results of sDNA operations can be displayed using layer styles. After running sDNA, right-click the relevant layer in the layers panel, choose `Properties` → `Style`, change `Single Symbol` to `Graduated` and select the data you want to display.

Autocad

When we originally created sDNA, we envisaged urban designers using it via Autocad. As sDNA has become more advanced, the data handling capabilities of Autocad no longer support all the features we offer; in particular, use of user data attached to links is not possible. We are not fixing this because the Urban Design world mostly uses BIM systems these days, and we plan to implement sDNA for BIM in future. If you are interested in this possibility, please get in touch with us!

If you are an **Autocad Map3d** user, there is a workflow for using fully featured sDNA models that involves exporting/importing data from the free QGIS. See our notes on *Advanced sDNA models in Autocad Map3d*.

For other products in the Autocad family, use of basic models (without user data) is supported. Installation is as follows:

1. On your start menu, in the sDNA program group, click on `Register sDNA for Autocad` (32 or 64 bit depending on your Windows installation). You may need to provide an administrator password.
2. From the Ribbon, choose `Manage` → `Load Application`

3. Under `Startup Suite` click `Contents...` then `Add...`
4. Navigate to the place where you installed sDNA (usually `c:\Program Files (x86)\sDNA`) and select the application *sdna.vlx*.
5. Click `Close` on the Startup Suite dialog
6. Click `Close` on the Load Application dialog
7. In Autocad versions 2010 onwards, load the sDNA buttons:
    a. From the Ribbon, choose `Manage` → `Custom User Interface (CUI)`
    b. Locate the button for loading a Partial Customization File – the icon is a folder symbol with a green plus sign
    c. Navigate to the place where you installed sDNA (usually `c:\Program Files (x86)\sDNA`) and select `sDNA.cuix`
    d. Click `OK`
8. Quit Autocad
9. When Autocad is restarted, the sDNA application will be loaded.

sDNA for Autocad processes networks of polylines using the coincident endpoint connectivity rule (see network preparation chapter for more details on this). Networks using an intersection-is-connectivity rule can be converted using the **sdnaconvertunlinks** tool. This will take a mixture of polylines and polygons as input, and automatically breaks all polylines where they intersect (unless the intersection falls inside a polygon, which you can use to denote brunels).

After running sDNA Integral, you will need to use the sDNA Colour tool to display results.

In Autocad 2010 onwards, sDNA will appear as a series of buttons on the ribbon toolbar labelled "sDNA". Simply click these buttons to load the tools.

In older versions of Autocad, it is necessary to know the commands for running sDNA. (These can also be used in Autocad 2010 onwards, if preferred). Enter one of the following at the command prompt:

- **sdnaconvertunlinks** to convert from intersection-connectivity to coincident endpoint connectivity
- **sdnaloaditn** to load ITN data
- **sdnaprepare** to prepare the network
- **sdnaintegral** to analyze the network
- **sdnacolor** or **sdnacolour** to display the results of **sdnaintegral**

Advanced sDNA models in Autocad Map3d

To use the full data capabilities of sDNA from Autocad Map3d, we recommend the following workflow:

1. Export data as a shapefile.
2. Process in the free *QGIS* or by *using sDNA from the command line*.
3. Re-import into Map3d.

This enables the use of Map3d's sophisticated 3d editing and snapping features in sDNA models. However, please take note of the following:

- Do not edit shapefiles as a Map3d mapping layer, as this discards 3d information.
- Instead, create your network using Autocad polylines.

- Models can be exported from Autocad polylines to shapefiles. Note that (1) it is necessary to manually specify export of all attached object data, (2) it is necessary to select the 3d export driver to preserve height data, and (3) care must be taken to preserve the spatial referencing.
- Shapefiles can be imported into Map3d as Autocad objects, with data attached as object data, and preserving spatial reference.

Using sDNA from the command line

sDNA can also be used to process shapefiles from the command line. Before starting, you will need to install Python, if you don't have it already. We have tested with versions 2.6 and 2.7; other versions may work as well. You can download Python 2.7.3 from here:

If your file associations are set up correctly for python (the python installer should have done this) and the sDNA bin directory (usually `c:\program files\sdna\bin`) has been added to your path (the sDNA installer should have done this), then you can use command line sDNA as follows.

The commands are `sdnaprepare.py`, `sdnaintegral.py`, `sdnalearn.py` and `sdnapredict.py`. Note that from the command line, various functions handled separately in QGIS and ArcGIS (geodesics, convex hulls, link measures, destination maps, network radii) are all handled by `sdnaintegral.py`. See *Advanced configuration and command line options* for more details; alternatively to learn the command for a given operation, try performing the operation from QGIS and see the command QGIS calls (shown in the algorithm dialog).

If you have ArcGIS 10.1 or later installed, then the command line interface to sDNA will also support work on geodatabase paths. Of course you can use sDNA from inside ArcGIS as well, but some of us like to make batch scripts to run outside of Arc.

Troubleshooting

If you haven't, can't or don't want to set up file associations, then the instructions above won't work. You will have to load python explicitly, e.g. (assuming python is on your system path):

```
python -u sdnaprepare.py --help
```

(or if it isn't):

```
c:\path\to\python -u sdnaprepare.py --help
```

(or if neither python nor the sdna bin folder are on your path)

```
c:\path\to\python -u c:\path\to\sdna\bin\sdnaprepare.py -help
```

and so on, for the other sDNA commands detailed above.

Using sDNA through the Python interface

Those experienced in programming may want to use sDNA Prepare and sDNA Integral directly through our Python API. This is called `sdnapy`; the canonical example of its use can be found in `runcalculation.py` in the sDNA program folder.

1.7 Guide to individual tools

1.7.1 Options common to multiple tools

Several tools share the following parameters:

- ***Input*, *Output*** Shape layers for input and output
- ***Start grade separation*, *End grade separation***

  Select fields for grade separation. If given, links will only be deemed connected if

  1. they share the same endpoints (x,y,z)
  2. they share the same grade separation value at those endpoints

  Grade separation can also be used to emulate multilayer/multimode networks. For example, use values of 0,1,2... for grade separation on roads, 10,11,12... for railways, etc.

- ***Analysis Metric*** Select metric for analysis: Euclidean, angular, custom, other (see *Distance Metric* for details).

  Custom metrics require selection of a custom metric data field.

  Some metrics provided are preset forms of hybrid metric designed for specific network types (pedestrian, vehicle, cycle, public transport). You can inspect the message output of sDNA to discover exactly what formula they use. Some presets include user variables, which have a default value but can be changed in advanced config; see *preset metric variables*.

- ***Weighting*, *Origin weight*, *Destination weight*** Select *Weighting* (link, length or polyline) and the data to use. If *origweightformula* or *destweightformula* are supplied in advanced config, they override origin and destination weight.
- ***Radii*** Enter one or more radii to use as radius of analysis (see *Radius* for details). By default these are Euclidean and expressed in the spatial units of the data, which should usually be metres if your data is projected correctly. If entering several, separate them by commas. Radius 'n' can be used to specify global analysis, limited only by the size of the network (but computationally intensive). Example: `100, 200, 400, 800, 1600, n`
- ***Continuous space*** Select whether *continuous space* analysis is used.
- ***Banded radius*** If this mode is chosen, measures for each radius exclude links contained in the next smallest radius. This allows multivariate analysis across radii while keeping cross-correlation of measures to a minimum.
- ***Disable lines*** Allows discarding of certain polylines from the analysis. This is useful for testing multiple design options, or for switching on and off parts of the network such as cycle paths.

  This parameter takes an *expression* which, if it evaluates to a nonzero result for a given polyline, will disable that polyline. In the simplest case, this can be a field name such as `walk_net` (which would disable all lines for which the walk_net field is nonzero) or a combination such as `walk_net||cycle_net` (which would disable both the walking and cycling network - useful for a vehicle analysis).

- *Origin Destination Matrix file*

  Optionally allows weights to be determined by an Origin-Destination (OD) matrix. The file must be formatted correctly, see *Creating a zone table or matrix file*. All geodesic and destination weights are replaced by values read from the matrix. The matrix is defined between sets of *zones*; polylines must contain fields to indicate their zone.

  In the case of sDNA Integral, *Two phase betweenness* is disabled, because use of a two phase model for determining geodesic and destination weights conflicts with use of an OD matrix to determine these.

- *Intermediate link filter*

  Optionally restricts the analysis to include only geodesics that pass through a given line or set of lines specified by the filter. Affects output of Betweenness, Two Phase Betweenness, Two Phase Destination, Mean Crow Flight, Mean Geodesic Length, Diversion Ratio measures; also Geodesic and Destination geometries.

This parameter takes an *expression* which, if it evaluates to a nonzero result for a given line, will include geodesics which pass through that line. In the simplest case, this can be a field name such as `my_link_filter` (which would disable all lines for which the field *my_link_filter* is nonzero).

It is not sufficient for a geodesic's origin or destination to pass the filter; an intermediate line must pass in order for the geodesic to be included.

- ***Advanced config*** Allows setting of parameters not shown in the interface. These are described in *advanced_config*.

1.7.2 Individual tool details

Preparation

Prepare network

Prepares spatial networks for analysis by checking and optionally repairing various kinds of error.

Note that the functions offered by sDNA prepare are only a small subset of those needed for preparing networks. A good understanding of *Network Preparation* is needed, and other (free) tools can complement sDNA Prepare.

The errors fixed by sDNA Prepare are:

- *endpoint near misses* (XY and Z tolerance specify how close a near miss)
- *duplicate lines*
- *traffic islands* (requires traffic island field set to 0 for no island and 1 for island). Traffic island lines are straightened; if doing so creates duplicate lines then these are removed.
- *split links*. Note that fixing split links is no longer necessary as of sDNA 3.0 so this is not done by default
- *isolated systems*

See *Options common to multiple tools*.

Optionally, numeric data can be preserved through a prepare operation by providing the desired field names, separated by commas, to the parameters *Absolute data to preserve* and *Unit length data to preserve*.

Individual Line Measures

Outputs connectivity, bearing, euclidean, angular and hybrid metrics for individual polylines.

This tool can be useful for checking and debugging spatial networks. In particular, connectivity output can reveal geometry errors.

See *Options common to multiple tools*.

Analysis

Integral Analysis

sDNA Integral is the core analysis tool of sDNA. It computes several flow, accessibility, severance and efficiency measures on networks. Full details of the analysis are given in *Analysis: what the results mean* and *Analysis: full specification*.

Integral allows output of various groups of measures to be switched on and off.

See *Options common to multiple tools*.

Specific Origin Accessibility Maps

Outputs accessibility maps for specific origins, including metric between each origin-destination, Euclidean path length and absolute diversion (difference between Euclidean path length and crow flight path length, similar to circuity, notated here as 'Div').

See *Options common to multiple tools*.

The accessibility map tool also allows a list of origin polyline IDs to be supplied (separated by commas). Leave this parameter blank to output maps for all origins.

If outputting "maps" for multiple origins, these will be output in the same feature class as overlapping polylines. It may be necessary to split the result by origin link ID in order to display results correctly.

Integral from OD Matrix (assignment model)

A simplified version of sDNA Integral geared towards use of an external Origin Destination matrix. Note that several other tools (including Integral) allow Origin Destination matrix input as well.

The file must be formatted correctly, see *Creating a zone table or matrix file*. All geodesic and destination weights are replaced by values read from the matrix. The matrix is defined between sets of *zones*; polylines must contain text fields to indicate their zone.

Skim Matrix

Skim Matrix outputs a table of inter-zonal mean distance (as defined by whichever sDNA Metric is chosen), allowing high spatial resolution sDNA models of accessibility to be fed into existing zone-base transport models.

Geometries

The geometry tools output individual geometries used in an integral analysis. These may be useful either for visualization, or for exporting to external analysis tools. For example, you could join geodesics to a pollution dataset to estimate exposure to pollution along everyday travel routes.

Convex Hulls

Outputs the convex hulls of network radii used in *Integral Analysis*.

See *Options common to multiple tools*.

The convex hulls tool also allows a list of origin polyline IDs to be supplied (separated by commas). Leave this parameter blank to output hulls for all origins.

Geodesics

Outputs the geodesics (shortest paths) used by *Integral Analysis*.

See *Options common to multiple tools*.

The geodesics tool also allows a list of origin and destination polyline IDs to be supplied (separated by commas). Leave the origin or destination parameter blank to output geodesics for all origins or destinations. (Caution: this can produce a very large amount of data).

Network Radii

Outputs the network radii used in *Integral Analysis*.

See *Options common to multiple tools*.

The network radii tool also allows a list of origin polyline IDs to be supplied (separated by commas). Leave this parameter blank to output radii for all origins.

Calibration

sDNA Learn and Predict provide a way to calibrate sDNA outputs against measured variables (flows, house prices, etc). Currently they offer bivariate regression with Box-Cox transformation. Multiple predictor variables (the outputs of sDNA) can be tested to see which gives the best cross-validated correlation with the target variable.

Learn

sDNA Learn selects the best model for predicting a target variable, then computes GEH and cross-validated R^2. If an output model file is set, the best model is saved and can be applied to fresh data using sDNA Predict.

Available methods for finding models are:

- *Single best variable* - performs bivariate regression of target against all variables and picks single predictor with best cross-validated fit
- *Multiple variables* - regularized multivariate lasso regression
- *All variables* - regularized multivariate ridge regression (may not use all variables, but will usually use more than lasso regression)

Candidate predictor variables can either be entered as field names separated by commas, or alternatively as a *regular expression*. The latter follows Python regex syntax. A wildcard is expressed as `.*`, thus, `Bt.*` would test all Betweenness variables (which in abbreviated form begin with *Bt*) for correlation with the target.

Box-Cox transformations can be disabled, and the parameters for cross-validation can be changed.

Weighting lambda weights data points by $\frac{y^\lambda}{y}$, where y is the target variable. Setting to 1 gives unweighted regression. Setting to around 0.7 can encourage selection of a model with better GEH statistic, when used with traffic count data. Setting to 0 is somewhat analagous to using a log link function to handle Poisson distributed residuals, while preserving the model structure as a linear sum of predictors. Depending on what you read, the literature can treat traffic count data as either normally or Poisson distributed, so something in between the two is probably safest.

Ridge and Lasso regression can cope with multicollinear predictor variables, as is common in spatial network models. The techniques can be interpreted as frequentist (adding a penalty term to prevent overfit); Bayesian (imposing a hyperprior on coefficient values); or a mild form of entropy maximization (that limits itself in the case of overspecified models). More generally it's a machine learning technique that is tuned using cross-validation. The r^2 values reported by learn are always cross-validated, giving a built-in test of effectiveness in making predictions.

Regularization Lambda allows manual input of the minimum and maximum values for regularization parameter λ in ridge and lasso regression. Enter two values separated by a comma. If this field is left blank, the software attempts to guess a suitable range, but is not always correct. If you are familiar with the theory of regularized regression you may wish to inpect a plot of cross validated r^2 against λ to see what is going on. The data to do this is saved with the output model file (if specified), with extension `.regcurve.csv`.

Predict

Predict takes an output model file from sDNA Learn, and applies it to fresh data. For example, suppose we wish to calibrate a traffic model, using measured traffic flows at a small number of points on the network.

- First run a Betweenness analysis at a number of radii using *Integral Analysis*.
- Use a GIS spatial join to join Betweenness variables (the output of Integral) to the measured traffic flows.
- Run *Learn* on the joined data to select the best variable for predicting flows (where measured).
- Run *Predict* on the output of Integral to estimate traffic flow for all unmeasured polylines.

1.7.3 Advanced configuration and command line options

sDNA supports a wide variety of options for customizing the analysis beyond what is shown in the user interface. All of these are accessed through the advanced config system.

Advanced config options are specified in a long string with options separated by semicolons (;) like this:

```
nohull;probroutethreshold=1.2;skipzeroweightorigins
```

This is an example of an advanced config for sDNA Integral, which means

- Don't compute convex hull
- Problem route threshold = 1.2
- Skip zero weight origins

When calling sDNA *Integral Analysis* and *Prepare Network* from the command line (*Using sDNA from the command line*), the entire configuration is specified as an advanced config. Therefore, the advanced config options include some which are usually set via the graphical interface. If these options are given as advanced config in the sDNA graphical interface, an error ("Keyword specified multiple times") will result.

Advanced config options for sDNA Prepare

Option	Description
startelev=	Name of field to read start elevation from
endelev=	Name of field to read end elevation from
island=	Name of field to read traffic island information from. Anything other than zero will be treated as traffic island
islandfieldstozero=	Specifies additional data fields to set to zero when fixing traffic islands (used for e.g. origin or destination weights)
data_unitlength=	Specifies numeric data to be preserved by sDNA prepare (preserves values per unit length, averages when merging links)
data_absolute=	Specifies numeric data to be preserved by sDNA prepare (preserves absolute values, sums when merging links)
data_text=	Specifies text data to be preserved (merges if identical, concatenates with semicolon otherwise)
xytol=	Manual override xy tolerance for fixing endpoint connectivity
ztol=	Manual override z tolerance for fixing endpoint connectivity
merge_if_identical=	Specifies data fields which can only be merged if identical, i.e. split links will not be fixed if they differ (similar to 'dissolve' GIS operation)

xytol and ztol are manual overrides for tolerance. sDNA, running on geodatabases from command line or ArcGIS, will read tolerance values from each feature class as appropriate. sDNA running in QGIS or on shapefiles will use a

default tolerance of 0, as shapefiles do not store tolerance information:- manual override is necessary to fix tolerance on shapefiles.

Advanced config options for sDNA Integral and geometry tools

sDNA Convex Hulls, Network Radii, Geodesics and Accessibility Map are all different interfaces applied to sDNA Integral, so will in some cases accept these options as well.

Option	Default	Description
startelev=		Name of field to read start elevation from
endelev=		Name of field to read end elevation from
metric=	angular	Metric – angular, euclidean, custom or one of the presets
radius=	n	List of radii separated by commas
startelev=		Name of field to read start elevation from
endelev=		Name of field to read end elevation from
origweight=		Name of field to read origin weight from
destweight=		Name of field to read destination weight from
origweightformula=		Expression for origin weight (overrides origweight)
destweightformula=		Expression for destination weight (overrides destweight)
weight=		Name of field to read weight from. Applies weight field to both origins and destinations.
zonesums=		Expressions to sum over zones (see zone sums below)
lenwt		Specifies that weight field is per unit length
custommetric=		Specified field name to read custom metric from
xytol=		Manual override xy tolerance for fixing endpoint connectivity.
ztol=		Manual override z tolerance for fixing endpoint connectivity.
outputgeodesics		Output geometry of all pairwise geodesics in analysis (careful – this can create a lot of data)
outputdestinations		Output geometry of all pairwise destinations in analysis (careful – this can create a lot of data). Useful in combination with origins for creating a map of distance/metric from a given origin.
outputhulls		Output geometry of all convex hulls in analysis
outputnetradii		Output geometry of all network radii in analysis
origins=		Only compute selected origins (provide feature IDs as comma separated list). Useful in conjunction with outputgeodesicsm, outputdestinations, outputhulls, outputnetradii.
destinations=		Only compute selected destinations (ditto)
nonetdata		Don't output any network data (used in conjunction with geometry outputs)
pre=		Prefix text of your choice to output column names
post=		Postfix text of your choice to output column names
nobetweenness		Don't calculate betweenness (saves a lot of time)
nojunctions		Don't calculate junction measures (saves time)
nohull		Don't calculate convex hull measures (saves time)
linkonly		Only calculate individual link measures.
outputsums		Output sum measures SAD, SCF etc as well as means MAD, MCF etc.
probroutes		Output measures of problem routes – routes which exceed the length of the radius
forcecontorigin		Force origin link to be handled in continuous space, even in a discrete analysis. Prevents odd results on very long links.
nqpdn=	1	Custom numerator power for NQPD equation
nqpdd=	1	Custom denominator power for NQPD equation
skipzeroweightorigins		Skips calculation of any output measures for origins with zero weight. Saves a lot of time if many such origins exist.
skipzeroweightdestinations	1	Zero weight destinations are skipped by default. Note this will exclude them from geometry outputs; if this is not desired behaviour then set skipzeroweightdestinations=0

Continued on next page

Table 2 – continued from previous page

Option	Default	Description
skiporigin	ifzero=	Specified field name. If this field is zero, the origin will be skipped. Allows full customization of skipping origins.
skipfraction	1	Set to value n, skips calculation for (n-1)/n origins. Effectively the increment value when looping over origins.
skipmod=	0	Chooses which origins are calculated if skipfraction?1. Effectively the initial value when looping over origins: every skipfractionth origin is computed starting with the skipmodth one.
nostrictnetwork	cut	Don't constrain geodesics to stay within radius. This will create a lot more 'problem routes'. Only alters behaviour of betweenness measures (not closeness).
probrouteaction	ignore	Take special action for problem routes that exceed the radius by a factor greater than probroutethreshold. Can be set to ignore, discard or reroute. Reroute changes geodesic to shortest Euclidean path. Only alters betweenness output, not closeness.
probroutethreshold=	1.2	Threshold over which probrouteaction is taken. Note this does not affect computation of probroutes measures, which report on all routes which exceed the radius length regardless of this setting.
outputdecomposableonly		Output only measures which are decomposable i.e. can be summed over different origins (useful for parallelization)
linkcentretypes=	Angular for angular analysis, Euclidean otherwise	Override link centre types – angular or Euclidean
lineformula=		Formula for line metric in hybrid analysis (see below)
juncformula=	0	Formula for junction turn metric in hybrid analysis (see below)
bidir		Output betweenness for each direction separately
oneway=		Specified field name to read one way data from (see note 1 below)
vertoneway=		Specified field name to read vertical one way data from (see note 1 below)
oversample=	1	Number of times to run the analysis; results given are the mean of all runs. Useful for sampling hybrid metrics with random components.
odmatrix		Read OD matrix from input tables (a 2d table must be present)
zonedist=	euc	Set expression to determine how zone weights are distributed over links in each zone, or 0 to skip distribution (all lines receive entire zone weight)
intermediates=		Set expression for intermediate link filter. Geodesics are discarded unless they pass through link where expression is nonzero.
disable=		Set expression to switch off links (links switched off when expression evaluates nonzero)
outputskim		Output skim matrix file
skimorigzone		Origin zone field (must be text) for skim matrix
skimdestzone		Destination zone field (must be text) for skim matrix
skimzone		Skim matrix zone field for both origin and destination (sets both skimorigzone and skimdestzone)
bandedradii		Divide radius into bands: for each radius only include links outside the previous radius
datatokeep=		List of field names for data to copy to output

Preset metric variables

A number of preset metrics are provided. These are special cases of hybrid metrics, sometimes with a fairly complex formula. To inspect the formula for a given metric, run *Individual Line Measures* with the metric selected, and inspect the message output where the full formula will be shown.

The CYCLE_ROUNDTRIP metric, as the name implies, measures a round trip to take account of hills in both directions.

Certain variables within the preset metric formulae can be changed by assigning to them in advanced config. To date, the list is:

Metric	Variable	Default	Meaning
Cycle	aadtfield	aadt	Name of data field containing annual average daily vehicle traffic estimate
Cycle	t	0.04	Tendency of cyclists to avoid vehicle traffic
Cycle	a	0.3	Tendency of cyclists to avoid angular change
Cycle	s	0.5	Tendency of cyclists to avoid slope

Interpretation of one way data

One way data is interpreted as follows:

- 0 – traversal allowed in both directions (so long as *vertoneway* allows this too)
- positive number – forward traversal only
- negative number – backward traversal only

Forwards/backwards are taken with respect to the direction in which the link is drawn in the network (ordering of points in the data).

Vertical one way data is interpreted as follows:

- 0 – traversal allowed in both directions (so long as *oneway* allows this too)
- positive number – upward traversal only
- negative number – downward traversal only

Upward/downward are deduced by measuring the endpoints of the link only. In the event that these have the same elevation/height and this leads to ambiguity, sDNA will print an error message and exit.

If conflicting *oneway* and *vertoneway* data are provided, sDNA will print an error message and exit. Note that if either field is zero, the other is permitted to override it without conflict.

Creating a zone table or matrix file

sDNA can read custom zone data, that is, data attached to *zones* rather than individual lines in the network. This can come from

- one-dimensional zone tables: provide the zone files to sDNA's inputs, and then reference the variables in expressions in the same way as you would use network data. This performs a function similar to a database join, to link zonal data to individual polylines. See *Zone Data and Zone Sums*.
- a custom origin-destination (OD) matrix: provide sDNA with a two-dimensional table and it will override all other weights

One dimensional tables can be provided in *list* format, and two dimensional tables can be provided in *list* or *matrix* format. The *list* format allows for sparse data, that is, data need not be given for all zones, and is assumed to be zero where not given.

All tables must be saved in CSV (comma separated) format.

1d table in list format

list	1		
zone	houses	jobs	schools
westeros	2000	4000	4
royston vasey	1800	7	1
mordor	600	10000	0
narnia	2100	500	3

A 1d table in list format must have

- *list* and *1* in the header row
- zone field name and data names in the second row. The network must contain a text field with name matching the zone field name (in this case "zone")
- zones and data below

2d table in list format

list	2	
zone	zone	flow
shire	rivendell	4
rivendell	mordor	6
mordor	shire	2
gondor	mordor	10000
mordor	gondor	5000

A 2d table in list format must have

- *list* and *2* in the header row
- origin and destination zone field names followed by data names in the second row. The network must contain a text field with name matching the zone field names. In this case, the origin and destination zones are drawn from the same set so these are both named "zone". Different sets of zones for origin and destination are supported however (e.g. for use with census residential and workplace zones).
- zones and data below

2d table in matrix format

matrix	zone	zone		
flow	shire	rivendell	gondor	mordor
shire	0	4	0	0
rivendell	0	0	0	6
gondor	0	0	0	10000
mordor	2	0	5000	0

This table shows the same data as the 2d table in list format above

A 2d table in matrix format must have

- *matrix* in the first line followed by the origin zone field name then the destination zone field name. The network must contain a text field with name matching the zone field names. In this case, the origin and destination zones are drawn from the same set so these are both named "zone". Different sets of zones for origin and destination are supported however (as with 2d list tables above).
- the second row starts with the name of the data, then the name of each destination zone
- the left column from row 3 downwards contains the name of each origin zone
- the remainder of the matrix contains the data

Zone Data and Zone Sums

Zone data is accessed in the same way as field data in expressions, described below. The following computes origin weights by multiplying *zoneweight* (taken from a table provided to sDNA) with the euclidean length of each polyline:

```
origweightformula = zoneweight * euc
```

Using *zonesums* in sDNA Integral's advanced config, it is possible to sum data over network zones. This is useful for controlling how zonal weights are distributed over polylines. The following example

- gives an example of how to use multiple zone schemes. It assumes two zonal variables are provided; *residential_weight* is defined for each zone in *res_zone*, and *retail_weight* is defined for each zone in *ret_zone*. In each case, the zone file will specify the fieldname which tells sDNA which zone each polyline belongs to.
- gives an example of how to compute multiple zone sums. These are specified in the form *sum1=expr1@zonefield1,sum2=expr2@zonefield2,...*. The config creates two zone sum fields, *eucsum* which is the total Euclidean length in each residential zone (*res_zone*), and *linksum* which is the total link count in each retail zone (*ret_zone*).
- gives an example of how to distribute zonal weights over the zones. *origweightformula* distributes the *residential_weight* zonal variable evenly over network length in each residential zone, while *destweightformula* distributes the *retail_weight* zonal variable evenly over links in each retail zone. (Note that polylines may constitute partial links, hence the use of *FULLlf*):

```
zonesums = eucsum=euc@origzonefield, linksum=FULLlf@destzonefield;␣
↪origweightformula = residential_weight*proportion(euc,eucsum);␣
↪destweightformula = retail_weight*proportion(FULLlf,linksum)
```

The `proportion(x,y)` function divides `x` by `y`, which is useful to work out what proportion of zone weight is found in the current link. It correctly handles the special cases where the zone contains no weight.

Note that *origweightformula* and *destweightformula* are always computed in discrete, rather than continuous space.

Expression reference

Operator (in reverse order of precedence)	Name	Example	Meaning
,	Statement separator	a,b,c	Do a, then b, then output c
=	Assignment	_a=b	Set _a equal to b
?:	If-then-else	p?x:y	If p then x else y
&&	Logical and	a&&b	a and b
||	Logical or	a||b	a or b
<=	Less than or equal	a<=b	a is less than or equal to b
>=	Greater than or equal	a>=b	a is greater than or equal to b
!=	Not equal	a!=b	a is not equal to b
==	Equal	a==b	a is equal to b
>	Greater than	a>b	a is greater than b
<	Less than	a<b	a is less than b
+	Addition	a+b	a plus b
-	Subtraction	a-b	a minus b
*	Multiplication	a*b	a times b
/	Division	a/b	a divided by b
^	Exponentiation	a^b	a to the power of b
()	Parentheses	2*(x+1)	add one to x then multiply by 2

Builtin functions	
sin(x), cos(x), tan(x) asin(x), acos(x), atan(x) sinh(x), cosh(x), tanh(x) asinh(x), acosh(x), atanh(x)	Trigonometric functions of x (in radians).
log2(x)	Logarithm of x base 2
log10(x), log(x)	Logarithm of x base 10
ln(x)	Logarithm of x base e
exp(x)	e to the power of x
sqrt(x)	Square root of x
sign(x)	-1 if x is negative, else 1
rint(x)	x rounded to nearest integer
abs(x)	Absolute value of x
min(a,b,c,...) max(a,b,c,...) sum(a,b,c,...) avg(a,b,c,...)	Minimum, maximum, sum and average of all arguments
trunc(x,l,u)	Truncate x to the range [l,u] (including endpoints)
randnorm(m,s)	Random number drawn from normal distribution with mean m and standard deviation s
randuni(l,u)	Random number drawn from uniform distribution on range [l,u]
proportion(x,y)	Divides x by y. Returns 0 if x=y=0 and stops calculation with error if x>0 and y=0. Useful for distributing zonal weights over links.

Random numbers are generated from Mersenne Twister mt19937 algorithm. *"Mersenne Twister: A 623-dimensionally equidistributed uniform pseudo-random number generator", Makoto Matsumoto and Takuji Nishimura, ACM Trans-*

actions on Modeling and Computer Simulation: Special Issue on Uniform Random Number Generation, Vol. 8, No. 1, January 1998, pp. 3-30.

Constants	
inf	infinity
pi	pi

Variables	
ang	Angular change
euc	Euclidean distance
hg	Height gain
hl	Height loss
FULLang	Angular change for entire polyline
FULLeuc	Euclidean distance for entire polyline
FULLhg	Height gain for entire polyline
FULLhl	Height loss for entire polyline
FULLlf	Link fraction for entire polyline
_x	*(where x is any name)*: Temporary variable (initialized to 0)
x	*(where x is any name not used as function or other value)*: field data on polyline

Any variable can be assigned to with =, but the new value will only affect the current formula being evaluated (assigning to ‘‘ang‘‘ will not change the shape of the network, for example!). It is recommended to use only temporary variables of the form ‘‘_x‘‘ as targets for assignment.

1.8 Step by step guides for specific tasks

1.8.1 Downloading and preparing data from OpenStreetMap

NOTE: AS OF 2020 THIS SECTION IS LARGELY OUT OF DATE. OSM DATA CAN BE CONNECTED USING THE bpol OPTION IN GRASS v.clean; THIS HANDLES ALL INTERSECTIONS CORRECTLY.

OSM is a crowd-sourced mapping product and as such is particularly well supported by the open source community. This tutorial therefore uses the free QGIS.

Numerous online services exist for extracting OSM data into shapefiles. One such service is Mapzen: Mapzen extracts come with more attribute data attached than most (including e.g. cycle routes) but are available only for urban areas. If you need larger scale or rural OSM extracts, you may find other services suit your purposes better; the OSMDownloader plugin available within QGIS can also be useful.

OSM contains numerous types of lines feature beside roads and paths – for example walls, fences and river banks. You will need to filter out the roads and paths only. This can be achieved using `Layer` → `Filter...`. Results of a layer filter can be saved as a new layer using `Layer` → `Save As...`; at the same time a new coordinate system can be chosen to ensure a suitable *Spatial reference* for sDNA analysis, and the saved layer can be restricted to selected features only, allowing extraction of the desired modelling area from the OSM download.

In Cardiff (as of November 2014) a road was present when the ‘highway’ field was set to one of the following: `living_street`, `motorway`, `motorway_link`, `primary`, `primary_link`, `residential`, `road`, `secondary`, `secondary_link`, `service`, `services`, `tertiary`, `tertiary_link`, `trunk`, `trunk_link`, `unclassified`. Note that other areas may differ, and these classes may change over time. We recommend examining all values which appear in the ‘highway’ field to decide for yourself which to use.

Cycle lane data (that is, presence of cycle lanes on roads) was not extensive enough to be useful, but data on cycle routes (major cycle routes through cities) was consistent. Cycle routes do not necessarily denote highways, but are

a separate feature that coincides with the highways used. Some of the marked cycle routes coincided with features which were classed as paths rather than highways (a data inconsistency). Therefore it was necessary to extract all candidate highways for cycle routes using the following procedure:

- Select all lines where `route="bicycle"`
- Create 5m buffers surrounding these lines
- Create a new data field on the OSM data, "on_cycle_route"
- Use spatial join to set on_cycle_route=1 for all OSM lines within the cycle route buffers
- Use the filter `highway IS NOT NULL AND on_cycle_route = 1` to extract all highways coinciding with cycle routes.

With further use of the `Filter` tool it is possible to create two fields on the extracted highway network:

- cycle_route – set to 1 if on a cycle route, 0 otherwise
- car_net – set to 1 if on car (road) network, 0 otherwise

Reducing such data to numeric values is important if it is to be preserved during the *Prepare network* process, or if filtering of links is desired from *Integral Analysis*.

As of November 2014, the OSM data for Cardiff also contained a number of connectivity and geometry errors. These were fixed by use of topology tools (ArcGIS `Planarize` or GRASS `v.clean.advanced`). OSM represents bridges and tunnels with non-intersected lines, so it is recommended to preserve this format.

1. Filter out bridges and tunnels (brunels), to avoid breaking them, using the SQL query:

   ```
   ("bridge" <> 'no' AND "bridge" <> ' ') or "tunnel" = 'yes'
   ```

   and save them to a new layer.

   It is strongly recommended to inspect all values which appear in the bridge and tunnel fields in order to be sure that you are capturing all information provided by OSM. In our case the bridge field could contain the values "", "no", "suspension", "viaduct", "swing", "transporter" and "yes" - if these are present in your data, modify the query to account for them.

2. Filter out all other links:

   ```
   NOT (("bridge" <> 'no' AND "bridge" <> ' ') or "tunnel" = 'yes')
   ```

   (modified for other bridge types if necessary) and save them to a new layer.

3. On the "other links" layer run `v.clean.advanced` from the GRASS toolbox. The GRASS tools are bundled with the free QGIS, though to display them it is necessary to switch the Processing toolbox to advanced mode. (This is necessary for *Using sDNA for the first time* in any case). For the `Cleaning tools` parameter, enter:

   ```
   snap,break,rmline
   ```

   and for `threshold` enter 1 (assuming a suitable coordinate system was chosen above, with units in metres). This will merge nearby lines, remove duplicates, break at intersections and remove zero length lines.

4. Re-merge the cleaned "other links" with the brunel layer using `Vector` → `Data Management` → `Merge Shapefiles to one...`

5. Run sDNA Line Measures on the merged data to compute connectivity. Set layer properties to label each feature with connectivity (LConn). Using the brunels layer as a reference, manually check that brunels have been correctly reconnected with ordinary links. (If the attribute table for the brunels layer is opened, each feature can be selected and zoomed to in turn, and the corresponding feature on the merged class checked).

1.8.2 Modelling a combined vehicle and cycle network

This tutorial is intended to illustrate the use of sDNA in practice. It deals with one of the most complex tasks we have achieved with sDNA to data; modelling a cycle network in which cyclists avoid motor vehicles, slope and twistiness. However, several lessons for other model types (vehicle and pedestrian) are illustrated by the same process, so the example is worthwhile reading whatever your intended end use of sDNA.

The tutorial is written with sDNA and ArcGIS in mind, though could apply to any GIS software. Steps specific to ArcGIS are written in *italics*. Some features (hybrid metrics, one way streets) are not available in standard sDNA, though the principles of modelling remain the same.

sDNA cycle models are based on detailed behavioural simulation and have various outputs including

1. Predicted vehicle flows
2. Predicted cyclist flows
3. Cycling accessibility – quantity and quality of provision
4. Effect of traffic and slope on accessibility
5. Roads with predicted conflicts between cyclist and motor vehicles
6. Maps of cycling flows to specific facilities
7. Maps of cycling accessibility to specific facilities
8. Quantification of change to accessibility caused by proposed scheme

All of these outputs can be used in two ways

1. For gap analysis, to identify areas of need prior to designing schemes
2. For evaluating proposed schemes

Additionally, the model can be based on network shape alone, or take existing land use / commuting patterns into account. The alternatives are

1. Land use weighted models. These allow for weighting origins and destinations within the network, e.g. homes and railway stations as origins, employers and retail as destinations.
2. Origin Destination (OD) matrix models (forthcoming in 2016). These allow input of an existing OD matrix, such as commuting flows from census data.
3. Network shape only models. These use only the network shape itself to predict flows.

Model types 1 and 2 are likely to have higher correlation with observed flows than type 3, however type 3 (network only) models are the easiest to produce and perhaps the most useful. Type 1 models incur the problem of determining suitable weights for different types of land use. Both type 1 and 2 models also make the assumption that existing land use and commuting patterns will remain unchanged. In the long term this is not true, as new developments occur and major employers open and close. Such factors are typically outside the control of those analysing the network in any case. It may be appropriate to model these factors in some cases, for example, if there is an explicit need to design transport infrastructure around an existing major employer. In most cases however we recommend working with network shape only; although the models will exhibit reduced fit to current data compared to other model types, they are likely to remain more applicable in the long term.

The modelling process includes the following steps:

1. Create a routable vehicle and cycle network
2. Run the vehicle network model
3. Calibrate the vehicle model
4. Produce a cycle model informed by the calibrated vehicle model

5. (Optional) Calibrate the cycle network model. If this step is omitted, the cycle model will not show predicted counts but will still show relative levels of flow, which may be all that is needed.
6. Produce desired outputs from the cycle model

Production of a routable network

This tutorial assumes a routable network accurate enough to use for modelling is already available. Producing such a network is usually the most time consuming part of any modelling exercise; fortunately in many cases this step has been completed already, or if not, then good material is already available to work with. OpenStreetMap for example contains mapping of vehicle and cycle links in some areas.

General considerations when preparing a network for analysis are discussed in *Network Preparation*. This is essential reading for all users of sDNA models. More specific notes on Open Street Map can be found in *OpenStreetMap (OSM)* and a step-by-step guide to its use in *Downloading and preparing data from OpenStreetMap*.

A special consideration applies when producing a vehicle and cycle model, namely that it must be possible to join (preferably with ease) the predicted flows from the vehicle model to the cycle network. The best way to do this is to produce a single input network for both models, with two attached data fields to show whether access to each link is permitted (i) by cars and (ii) by cycles. Results from different model runs can then be joined easily by either geometry (spatial join) or object ID.

Table 1 shows an example of a unified network format. In it, link 1 and 4 are ordinary roads traversable by both vehicles and cycles, link 2 would be a motorway accessible only to vehicles, and link 3 a (short) section of traffic free cycle path. Links 2 and 4 have one way systems in place, and link 2 ends on a bridge (shown by the grade separation field end_gs – to be discussed below). Link 5 is not a real network link but represents a city outside the model containing a million links. The "start" and "end" of the grade separation fields, and also the one way data, refer to the direction in which the link is drawn in the graphics program or GIS rather than any "natural" direction it has.

Object ID	Shape length	carnet	bikenet	onewaydata	weight	start_gs	end_gs
1	42.6	1	1	0	1	0	0
2	89	1	0	1	1	0	1
3	2	0	1	0	1	0	0
4	8.4	1	1	-1	1	0	0
5	1	1	1	0	1000000	0	0

Table 1. Sample excerpt from combined cycle and vehicle network

If the road network data encodes dual carriageways as separate links (that is, each half of the dual carriageway is represented by its own link in the network) then it is essential to ensure simulated vehicle traffic makes use of both sides of the carriageway. If this is not done, all vehicle traffic will take the most convenient side, leaving the other as an apparently useful traffic free cycle route. Obviously this does not happen in reality, because dual carriageways carry an explicit or implicit one way traffic restriction per side. The most reliable way to ensure correct usage of dual carriageways is therefore to input one way information into sDNA[1]. Table 2 shows how sDNA interprets one way data.

One way data	Meaning
0	Traversal in both directions allowed
1	Forwards traversal only
-1	Backwards traversal only

Table 2. sDNA encoding of one way data. "Forwards" and "backwards" relate to the direction in which the link is drawn in the graphics program or GIS.

[1] An alternative is to use a hybrid metric for vehicle route choice that includes a random component, so that vehicle routes are distributed between both sides of the carriageway. This may prove useful in cases where one way data is not available.

A third consideration in producing combined vehicle/cycle models is that vehicle network information will probably be needed for a much wider area than cycle network information. Table 3 gives an example. If we wish to model cycle flows within a city, we must make the cycle model slightly larger than the city itself to correctly model flows over the boundary in and out of the city. The vehicle model must in turn be quite a lot larger, in order to deduce where the medium distance traffic is travelling from and to. Although most vehicle trips are short, longer trips tend to aggregate to major through routes, so if we want to estimate the level of traffic on a major through route such as a motorway then we must also include some trips of at least medium length.

The best size of model to use will vary depending on travel behaviour in an area, but fortunately, spatial network models are not overly sensitive to the exact model size used. This is because network problems have a high level of *collinearity*, e.g. a link that is useful for 50km trips is also likely to be useful for 30km trips, so both types of model will be able to identify realistic patterns of flow.

Area	Radius from model centre (example)
Desired modelling area e.g. city-wide model	8km, as measured along network
Cycle network model	12km, as measured along network
Vehicle network model	30km, as measured along network

Table 3. Example of nested models

As the vehicle model we need is larger than the cycle model, note that in a unified model *we do not need cycle route information for the wide scale vehicle network.* Also, at greater distances from the study area, there is even a reduced need for accuracy in the vehicle network. It is included in the model purely as a source and sink for traffic, and so long as the major routes to major neighbouring towns/cities are included, the model is sufficient. This can be checked after running the model by inspecting the predicted flows to these places.

A fourth consideration is modelling of elevation and grade separation in the network.

1. Elevation measures the height of each point above ground. Use of elevation data makes cycle models more accurate as slope affects cyclists behavioural choices. sDNA reads network features in 3d, which therefore includes elevation if present. Typically, network data is downloaded, corrected and prepared in 2d form then *draped* over a 3d terrain model to produce a 3d network. (In ArcGIS: *ArcToolbox* → *3d Analyst* → *Functional Surface* → *Interpolate Shape*).

2. Grade separation relates to points where links cross, such as bridges and tunnels. (These are collectively referred to as brunels). Although in theory this is determined by elevation, the available elevation data may not be preciseWhether enough to distinguish different levels of a brunel, particularly if data has been obtained by draping over a terrain model. sDNA will thus accept grade separation data in addition to elevation.

   Whether or not it is necessary to give sDNA grade separation data depends on how brunels are geometrically encoded. In some network data, links which pass over or under one another are represented by lines which intersect but are not noded; in these cases grade separation is not needed. More commonly however, a node is placed at such points even though there is no connection between the links; in the latter case, grade separation data is needed to inform sDNA that there is no connectivity. *Network Preparation* discusses this topic in more detail.

A final consideration with the wider vehicle network model is whether to model the wide network in detail. Such modelling is computationally expensive and may not be needed. In some cases, e.g. where a neighbouring city is accessed by only one route, the entire neighbouring city can be discarded from the model and replaced with a single *weighted* network link. If this is the case, the weighted link should be added to the network at the same distance from the model area as the city it replaces. The special link would be weighted by the number of links in the city it replaced, while all normal links would be assigned a weight of 1. An example was shown in Table 1.

Running the vehicle network model

To run the vehicle network model, follow these steps.

1. Decide on a set of radii (maximum trip lengths) to model. For example, 15, 20, 25, and 30 kilometres. As discussed above, the exact lengths are not important, so much as having a range of sensible values which will later be tested for their fit to actual vehicle flows.

2. Configure sDNA integral. Figure 10 shows an example. The type of analysis chosen is "ANGULAR". This means that all vehicle traffic attempts to take the straightest route, which is a good approximation to driver behaviour. Radii are expressed in metres (assuming the underlying data is projected to a coordinate system measured in metres – see *Network Preparation* for more detail). Weights are taken from the "weight" field to allow modelling of distant cities efficiency. Links not in the car network are disabled by putting `!carnet` (the opposite of the carnet field) into the "disable links" expression. One way information is taken from the *oneway* field.

   UPDATE We now recommend selecting "Banded radius". This means each radius will represent only trips in the relevant distance band (e.g. 20-25km) rather than all trips below the band (0-25km). This helps with multivariate modelling.

3. Run sDNA Integral. For a 50km urban network this will likely take a few hours to run. *At time of going to press, a bug in ArcGIS prevents progress information being displayed if the model runs in the background. There are two workarounds to this. Either use sDNA from the command line or QGIS, or disable background processing in Geoprocessing → Geoprocessing options.*

4. Display the model results. The parameter of interest is Angular Betweenness (BtA) at each radius. At this time, simply check that the results look like a sensible flow map, with major roads identified as having high levels of flow.

Fig. 10: sDNA Integral configuration for vehicle model.

Calibrating the vehicle model

Angular Betweenness output from the vehicle model represents the level of predicted flow for each maximum trip length, so shows relative traffic levels on the roads in the model. To calibrate, we compare Betweenness with actual traffic flow data to achieve two things; (i) determine which radius of Angular Betweenness best fits actual flows; (ii) determining the multiplier (coefficient) that will convert this radius of Betweenness into an actual count of, say, annual average daily traffic (AADT).

The first requirement for calibration is to get hold of some vehicle flow data. In the UK this is freely downloadable from the Department for Transport[2]. Vehicle flow data may come in the form of a table that includes grid references for each count point; in this case it will need converting to a spatial format to display (*In ArcGIS File → Add Data |rarr| Add XY Data*). The vehicle data can then be cropped to the model area. It is only necessary to use vehicle points from within the cycle network model – those that fall in the extended vehicle network can be discarded. Obviously the more points are available, the better the calibration can be, but models can be effectively calibrated using fewer than 50 points.

The points must then be checked, and possibly adjusted, so that it is clear which link in the model each point is supposed to attach to. Depending on your network geometry, the recorded points will probably not fall exactly on links in the model. It is also often the case that vehicle flow data reports the flow on dual carriageways as the total flow

[2] Note that the DfT release major road and minor road data separately. For effective calibration it is necessary to download both data sets and merge them.

in both directions, rather than counting each side of the carriageway separately; in these cases therefore a recorded point relates to two links rather than one. The aim of the adjustment process is to prepare the data such that the *Join* function of any GIS will be able to link each flow to the correct link(s). Points which are ambiguous (i.e. it is not certain which link they relate to) must be discarded.

If the model requires summing flows from both halves of a dual carriageway, it is advisable to encode measured flows as *gates* rather than single points. A gate is a line drawn so as to intersect all the links it measures (and only links it measures). It is relatively easy to iterate through the set of flow data points, and create by hand a new feature class containing lines that fulfil this purpose; a GIS spatial join function can then be used to join each line (gate) to the nearest recorded flow point.

Once a correct and unambiguous set of vehicle gates has been created, these are again *joined* to the output from the vehicle model. The resulting data should be the same set of vehicle gates, augmented by the network data from the link(s) that each gate intersects. This means using the gates as the primary layer for the *join* (otherwise we would end up with the entire network augmented by the vehicle gate data – a much larger dataset and not what is wanted).

The process of calibration can then proceed. We now do this by combining predictions in whichever way best predicts flows using *multivariate ridge regression*.

Regression fits a line to the plot of flows against predictions, as shown in Figure 2. We measure how well the regression has worked using *proportion of variance explained*, or r^2. r^2 can range from 0 (for a useless model) to 1 (for a perfect model). High r^2 is preferable, but low r^2 does not necessarily indicate a bad model: it can also be caused by errors in measurement, especially with cycle flows. Even if there is a lot of variability the model cannot predict, the component that can be predicted can form a useful basis for decision making.

Vehicle traffic flows on roads (real or predicted) do not have a normal distribution. That is to say, most of the traffic is actually carried by a small subset of roads. This means we should not weight all data points equally. Additionally, transport planners do not weight data points equally, instead using a statistic called GEH to balance absolute and relative errors.

To account for this, we weight data points by setting lambda to a value between 0 (weight to minimize relative errors) and 1 (weight to minimize absolute errors). In practice, values around 0.7 tend to minimize GEH.

If running a bivariate model (SINGLE_BEST) we can choose instead to Box-Cox transform variables to achieve the same thing, but this is not recommended for multivariate models (MULTIPLE_VARIABLES or ALL_VARIABLES).

1. *Join* the vehicle gates to the network (*Right click vehicle gates layer in table of contents; Joins and relates, Join…, Join data from another layer based on spatial location*).
2. Load sDNA Learn and configure it as in Figure 11. The output model is to be saved at `d:\\veh_model`. The target variable is `aadt` (annual average daily traffic). Note use of a regular expression `.*BtA.*` to select ALL angular betweenness variables as potential predictors at once.
3. Inspect the outputs of sDNA Learn. Which variables were used and with what weighting? Is r^2 good enough? If so then jump to step 5.
4. If r^2 is not good enough, inspect residuals output by sDNA Learn: plot a graph of actual flow against prediction *(View → Graphs → Create graph).* (*These can be selected on the graph to locate them on the map).*

   Try to figure out where errors come from. They are likely to arise

   > (a) from network errors; in which case it is worth checking why traffic is not taking the route you expect - perhaps some links are not connected when they should be?
   >
   > (b) because some behaviour that happens in reality is not accounted for in the model; e.g. flows to town centre or some specific facility, people navigating by means other than most direct route. If desired, run extra models and combine with the existing ones before running sDNA Learn on the outputs of all models together.

   See also *Troubleshooting Models*.

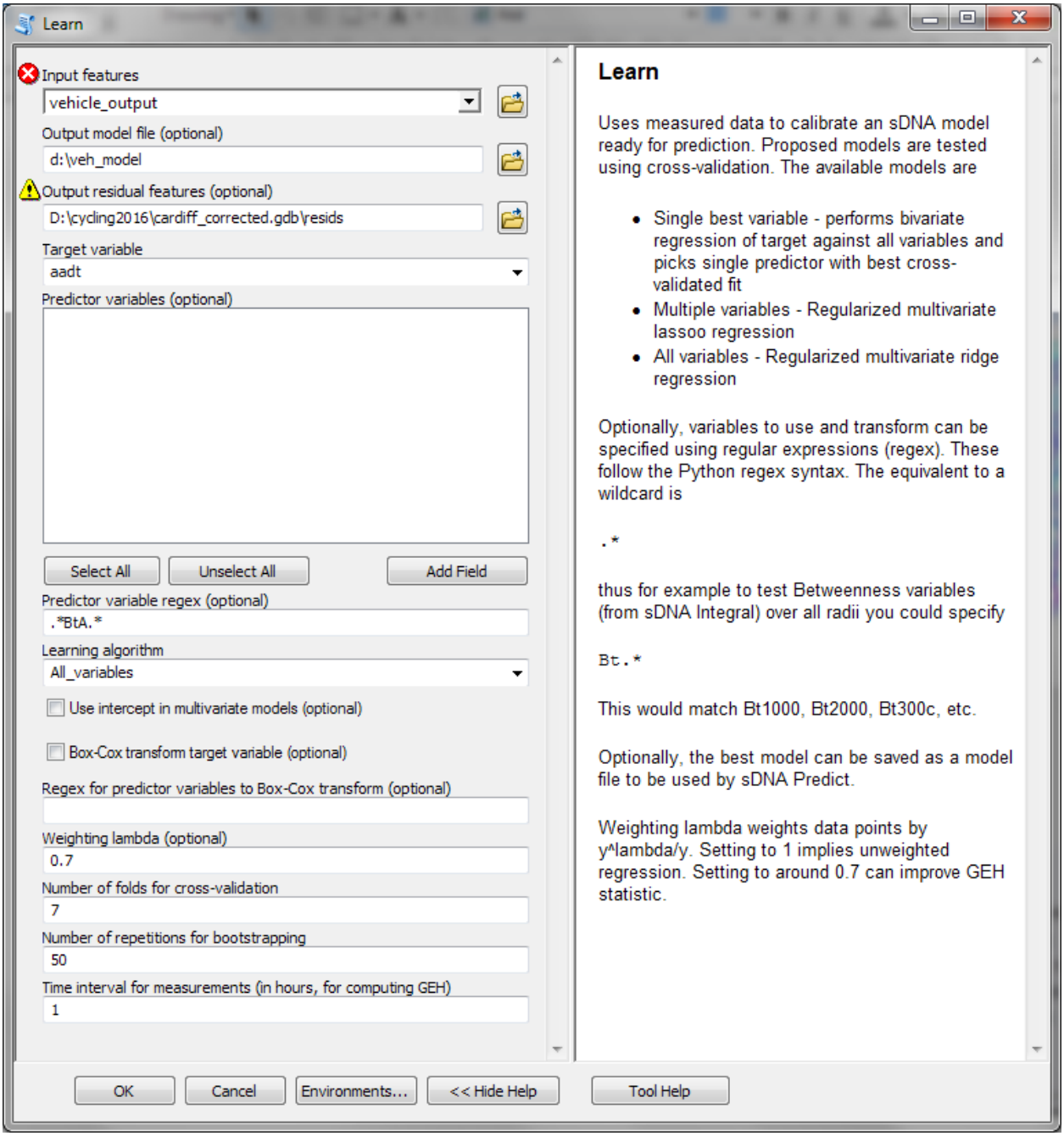

Fig. 11: sDNA Learn configuration for calibrating vehicle model.

5. Once you are happy with r^2, use sDNA Predict to predict aadt on the whole network, based on the model file (`d:\veh_model`). Note that if your vehicle gates consist of lines rather than points, the variable names must be changed slightly. For example, a variable called `Sum_BtA25000` on the *gates* would be called `BtA25000` on the *network*. The model file is a plain text file and can be edited to rename the variables as appropriate.

Calibration is now complete and the AADT field contains traffic predictions for each link in the combined network.

Production of the cycle model

Now that a calibrated estimate of vehicle traffic is present on the entire network, it is possible to produce a cycle model that takes account of vehicle traffic.

If using a unified network, we can now take a copy of only the desired study area plus buffer to account for cycle flows in and out of the area. Most cycle trips are not likely to be more than 4km in length, so this means the study area plus 4km (though a larger distance can be used if desired). Although distances should be measured along the network, it is fine to use simple selection tools and measure these distances as the crow flies – the result will only be that a slightly larger network is used. The actual trip lengths in the simulation are determined by the radius requested of sDNA.

It is worth checking at this point that elevation data has been correctly transferred to the network. One way to do this is to run sDNA Individual Link Measures, with a hybrid metric that captures slope.

1. Run sDNA Individual Link Measures with analysis type set to HYBRID, and the following advanced config: `linkformula=hg+hl`
2. Check that Hybrid Metric Forward (HmF) in the results from (1) highlights network links which slope.
3. Run a cycle model. This is again achieved by running sDNA Integral with the analysis type set to CYCLE_ROUNDTRIP. Set the Radial Metric to MATCH_ANALYTICAL: this means the radius is measured as CYCLE_ROUNDTRIP as well.

   Choose a variety of radii e.g. 3000, 4000, 5000, 7000, 9000, 12000, 15000, 18000. These are distances in metres adjusted for factors that deter cyclists: slope, traffic and twistiness. Note that CYCLE_ROUNDTRIP uses round trip distances to account for the fact that going downhill is a deterrant if the cyclist must come back up the hill later.

   Again, choose *Banded Radius*.

The sDNA cycle modes are derived from prior research on cyclist behaviour. It is a special case of hybrid analysis. If you inspect the output of sDNA, the equivalent hybrid formula is shown.

Calibration of the cycle model

The cycle model can be calibrated in much the same way as the vehicle one.

If no cycle flow data is available, then an uncalibrated model can be obtained by guessing at which radius of hybrid betweenness to use for the output. The most effective radius from cities similar to the study area would be a reasonable starting point.

If cycle flow data is available, then the best radius is selected using sDNA Learn in the same way as it was for the vehicle models. Having done this, it may or may not be necessary to convert the chosen betweenness variable into a predicted daily flow. The betweenness variable will already show relative levels of flow.

If predicted flows don't seem to match actual flows, see *Troubleshooting Models*.

Recommended model outputs

Predicted vehicle flows

These are present on the vehicle network, labelled "aadt" (or whatever you chose to call them during calibration).

Predicted cyclist flows

Uncalibrated flows are available as Betweenness Hybrid for varying maximum trip distances. Calibrated flows will be in a variable of your own creation if the calibration process is followed.

Accessibility

Accessibility is measured by computing network quantity within each distance band. It can thus be measured by the Links, Length or Weight variables, so long as the radial metric is set appropriately (e.g. CYCLE_ROUNDTRIP). If the model is calibrated, you have a few possible approaches

(a) quick and approximate: run a cycle model *without* using banded radius. Use sDNA learn in SINGLE_BEST mode to see which *single* radius best explains cyclist flows, and then measure accessibility for this radius.

(b) more sophisticated: keep the cycle model based on banded radius. Edit the model file, replacing BtH with Links, Length or Weight as appropriate, then use sDNA Predict to combine accessibility measures across multiple radii in the manner that best explains cyclist flows.

(c) If mode choice data is available, use sDNA Learn and Predict on the banded cycle model to produce predictions of mode choice. Use predicted mode choice as your definition of accessibility.

Higher numbers show higher quantity of network.

Effect of traffic and slope on accessibility

This is measured by comparing accessibility from the cycle model with an alternative model with no slope or traffic.

First add a numeric field to the network called "zero" and set it to 0 for all links. This is used to simulate zero traffic!

Run a second cycle model as described above but provide the following advanced config: `aadt=zero;s=0;pre=no_slope_traffic_`

`s=0` sets the effect of slope to 0. `aadt=zero` tells sDNA to use the contents of your "zero" field (which you set to 0) as the traffic estimate.

`pre=no_slope_traffic_` changes names for the outputs of the second model for clarity.

Now join the output of this second model back to the original model. Add and compute a data field to work out accessibility without traffic and slope/accessibility with traffic and slope. (ArcGIS: computing a field is not necessary. Instead, in the Symbology dialog, select accessibility without traffic/slope as Value and accessibility with traffic/slope as Normalization).

The results are expressed as a ratio, where 1 means traffic and slope have no effect, 2 means traffic and slope make accessibility twice as difficult, etc.

Note that producing flow predictions in the absence of traffic is also possible, and comparing these to flow predictions with traffic can also suggest hotspots to consider for infrastructure improvement.

Effect of traffic on accessibility

As *Effect of traffic and slope on accessibility*, but use a different advanced config: `pre=no_traffic_; aadt=zero`

Predicted conflicts

Create a variable that multiples predicted cycle flows by predicted vehicle flows. *(ArcGIS: use Add Field and Calculate Field in ArcToolbox - Data Management - Fields)*. The numbers are an unscaled score indicating relative risk of conflict. Display using only two categories, manually selecting a suitable threshold to identify the links with more conflict potential. The threshold should be set according to need; in terms of identifying incident sites:

- High thresholds increase the rate of both true and false negatives
- Low thresholds increase the rate of both true and false positives

Flows to/from specific facilities

Run a cycle model again, but weighted by the specific facility. This means creating a weight field on the network to denote the facility.

1. Add a field named station, hospital, etc
2. Set this field to 0 everywhere except for the facility site where it should be set to 1 *(ArcGIS: add field, calculate field for 0, then edit the facility itself to set field to 1)*.
3. Run a cycle model with analysis type set to CYCLE_ROUNDTRIP and destination weight set to the field created in (1).
4. Display Betweenness Hybrid for the appropriate radius

Accessibility to/from specific facilities

As for *Flows to/from specific facilities*, but display Mean Hybrid Distance (MHD) for the appropriate radius.

Change to accessibility caused by proposed scheme

This is measured by comparing accessibility as described above, from the current cycle model with an alternative model including the proposed scheme. Run a second cycle model that includes the new scheme and use GIS Spatial Join to bring data from the two models together. It may help to add `pre=before_` or `pre=after_` to advanced config to label the model outputs differently.

This value can be mapped to show improvement in accessibility, i.e. 1.23 = 23% more destinations to access for the same effort. Alternatively, it can be summed over the entire model to produce a score representing both the increase in accessibility, and the quantity of network that experiences the improvement.

Displaying model outputs

Recommended outputs from the model are discussed below.

We recommend using “graduated colours” symbologies, but modifying the highest bands of any key to display with thicker lines for extra emphasis.

- For flows, we recommend Geometrical Interval classification to give equal emphasis of high and low flows. This is to reflect the fact that flows tend to be exponentially distributed, i.e. most links have little flow but a few links have very high flow.
- It is also possible to display flows using Quantile or Jenks classification. This will condense bands of higher flows and provide more discernment of low level flows.
- For accessibility measures, we recommend Quantile or Jenks classification
- In some cases, e.g. ratios of accessibility, using manual classification may be useful, e.g. to create a special category for ratios equal to 1. A quantile classification can make a useful starting point for a manual one.

In ArcGIS, data on networks is displayed by right-clicking on a layer in the table of contents, → properties → symbology. For any display, click "Classify" from the symbologies dialog to determine how continuous data such as predicted flows is banded into categories.

With larger data sets, ArcGIS will warn that not all data has been sampled to create the classification. This is ill advised as some high-flow links may then fail to display altogether. To ensure all data is sampled, click "Classify" to set a higher number of features to sample.

Bibliography

[OSM1] See https://www.mapbox.com/osm-data-report/ and http://radar.oreilly.com/2014/08/ten-years-of-openstreetmap.html

[OSM2] http://wiki.openstreetmap.org/wiki/Research