# stplanr: A Package for Geographic Transport Data Science

Robin Lovelace *[1] and Richard Ellison[2]

[1]Institute for Transport Studies, University of Leeds

December 13, 2016

**Summary**

Tools for transport planning should be flexible, scalable and well-integrated with geographic data. **stplanr** meets each of these criteria by providing functionality commonly needed for transport planning in R, with an emphasis on spatial transport data. This includes tools to import and clean transport datasets; the creation of geographic 'desire lines' from origin-destination data; methods to assign these desire lines to the transport network, e.g. via interfaces to routing services such as CycleStreets.net, Graphhopper and the OpenStreetMap Routing Machine (OSRM); functions to calculate the geographic attributes of such routes, such as their bearing and equidistant surroundings; and 'travel watershed' analysis. With reproducible examples and using real transport datasets, this presentation will demonstrate how R can form the basis of a reproducible and flexible transport planning workflow. It concludes with a brief discussion of desirable directions of future development and ideas about functions for automating the generation of spatial interaction models.

**KEYWORDS:** transport planning, geographic data science, software, spatial interaction models.

## 1 Introduction

Transport planning is a diverse field requiring a wide range of computational tasks. Software in the transport planner's toolkit should be flexible, able to handle a wide range of data formats; robust, able to generate reproducible results for transparent decision-making; and scalable, able to work at multiple geographic levels from single streets to large cities and regions.

R can provide a solid basis for a transport planning workflow that meets each of these criteria.

Inspired by such efforts and driven by our own research needs, our primary aim for **stplanr** is to provide an R toolbox for transport planning (Lovelace and Ellison 2017). Although the focus is on spatial transport datasets (and most transport problems contain a spatial component), **stplanr** also provides functions for handling non-spatial datasets.

*r.lovelace@leeds.ac.uk

The design of the R language, with its emphasis on flexibility, data processing and statistical modelling, suggests it can provide a powerful environment for transport planning research. There are many quantitative methods in transport planning (Ortzar and Willumsen 2001) and we have attempted to focus on those that are most generalisable and frequently used. **stplanr** facilitates the following common computational tasks for transport planning:

- Accessing and processing of data on transport infrastructure and behaviour
- Analysis and visualisation of the transport network
- Analysis of origin-destination (OD) data and the visualisation of resulting 'desire lines'
- The allocation of desire lines to roads and other guideways via routing algorithms to show commonly used routes through geographical space
- The aggregation of routes to estimate total levels of flow on segments throughout the transport network
- Development of models to estimate transport behaviour currently and under various scenarios of change
- The calculation of 'catchment areas' affected by transport infrastructure

## 2 Package structure and functionality

The package can be installed and loaded in the usual way (see the package's README for dependencies and access to development versions):

```r
install.packages("stplanr")
```

```r
library(stplanr)
```

```
## Loading required package: sp
```

As illustrated by the message emitted when **stplanr** is loaded, it depends on **sp**. This means that the spatial data classes commonly used in the package will work with generic R functions such as `summary`, `aggregate` and, as illustrated in the figures below, `plot`.

### 2.1 Core functions and classes

The package's core functions are structured around 3 common types of spatial transport data:

- Origin-destination (OD) data, which report the number of people travelling between origin-destination pairs. This type of data is not explicitly spatial (OD datasets are usually represented as data frames) but represents movement over space between points in geographical space. An example is provided in the `flow` dataset.
- Line data, one dimensional linear features on the surface of the Earth. These are typically stored as a `SpatialLinesDataFrame`.
- Route data are special types of lines which have been allocated to the transport network. Routes typically result from the allocation of a straight 'desire line' allocated to the route

network with a `route_` function. Route network represent many overlapping routes. All are typically stored as `SpatialLinesDataFrame`.

For ease of use, functions focussed on each data type have been developed with names prefixed with `od_`, `line_` and `route_` respectively. Additional 'core functions' could be developed, such as those prefixed with `rn_` (for working with route network data) and `g_` functions for geographic operations such as buffer creation on lat/lon projected data (this function is currently named `buff_geo`). We plan to elicit feedback on such changes before implementing them.

All functions provided by **stplanr** can be printed with the `lsf.str("package:stplanr", all = TRUE)`.

## 2.2 Accessing and processing transport data

**stplanr** provides a variety of different functions that facilitate importing common data formats used for transport analysis into R. Although transport analysis generally requires some transport-specific datasets, it also typically relies heavily on common sources of data including census data as described in the package's vignette.

## 2.3 Creating geographic desire lines

Perhaps the most common type of aggregate-level transport information is origin-destination ('OD') data. This can be presented either as a matrix or (more commonly) a long table of OD pairs. An example of this type of raw data is provided below (see `?flow` to see how this dataset was created).

```
data("flow", package = "stplanr")
head(flow[c(1:3, 12)])
```

```
##        Area.of.residence Area.of.workplace All Bicycle
## 920573         E02002361         E02002361 109       2
## 920575         E02002361         E02002363  38       0
## 920578         E02002361         E02002367  10       0
## 920582         E02002361         E02002371  44       3
## 920587         E02002361         E02002377  34       0
## 920591         E02002361         E02002382   7       0
```

```
data("cents", package = "stplanr")
as.data.frame(cents[1:3, -c(3,4)])
```

```
##       geo_code  MSOA11NM coords.x1 coords.x2
## 1708 E02002384 Leeds 055 -1.546463  53.80952
## 1712 E02002382 Leeds 053 -1.511861  53.81161
## 1805 E02002393 Leeds 064 -1.524205  53.80410
```

We use od2line to combine flow and cents, to join the former to the latter. We will visualise the l object created below in the next section.

```
l <- od2line(flow = flow, zones = cents)
```

The data is now in a form that is much easier to analyse. We can plot the data with the command plot(l), which was not possible before. Because the SpatialLinesDataFrame object also contains data per line, it also helps with visualisation of the flows, as illustrated in Figure 1.

## 2.4 Allocating flows to the transport network

**stplanr** tackles this issue of the computational intensity of complex routing algorithms that can scale nationally and globally by using 3rd party APIs.

This is illustrated below with route_cyclestreet, which uses CycleStreets.net API, a routing service "by cyclists for cyclists" that offers a range route strategies (primarily 'fastest', 'quietest' and 'balanced') that are based on a detailed analysis of cyclist wayfinding:[1]

```
route_bl <- route_cyclestreet(from = "Bradford", to = "Leeds")
route_c1_c2 <- route_cyclestreet(cents[1,], cents[2,])
```

The raw output from routing APIs is usually provided as a JSON or GeoJSON text string. By default, route_cyclestreet saves a number of key variables (including length, time, hilliness and busyness variables generated by CycleStreets.net) from the attribute data provided by the API. If the user wants to save the raw output, the save_raw argument can be used:

```
route_bl_raw <- route_cyclestreet(from = "Bradford", to = "Leeds", save_raw = TRUE)
```

Additional arguments taken by the route_ functions depend on the routing function in question. By changing the plan argument of route_cyclestreet to fastest, quietest or balanced, for example, routes favouring speed, quietness or a balance between speed and quietness will be saved, respectively.

To automate the creation of route-allocated lines over many desire lines, the line2route function loops over each line, wrapping any route_ function as an input. The output is a SpatialLinesDataFrame with the same number of dimensions as the input dataset (see the right panel in Figure 1).

```
routes_fast <- line2route(l = l, route_fun = route_cyclestreet)
```

The result of this 'batch routing' exercise is illustrated in Figure 1. The red lines in the left hand panel are very different from the hypothetical straight 'desire lines' often used in transport research, highlighting the importance of this route-allocation functionality.

[1] An API key is needed for this function to work. This can be requested (or purchased for large scale routing) from cyclestreets.net/api/apply. See ?route_cyclestreet for details. Thanks to Martin Lucas-Smith and Simon Nuttall for making this possible.

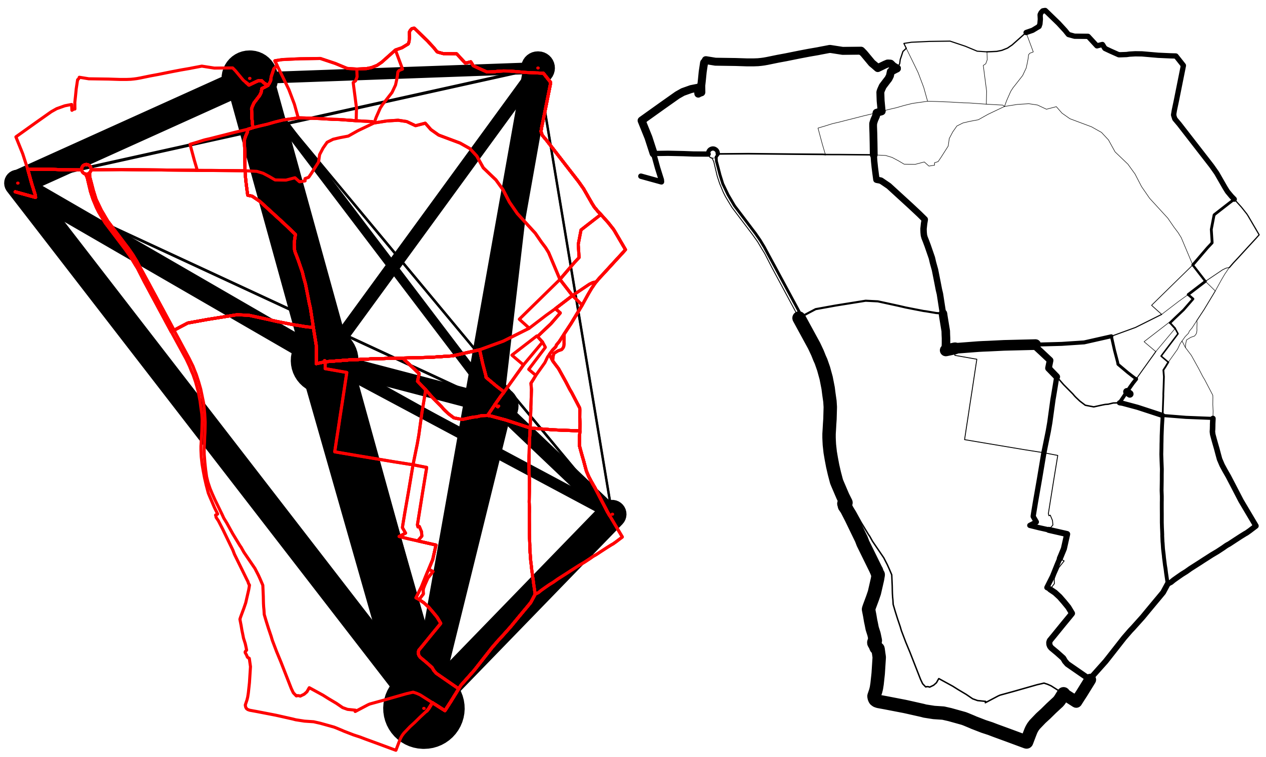

Figure 1: Visualisation of travel desire lines, with width proportional to number of trips between origin and destination (black) and routes allocated to network (red) in the left-hand panel. The right hand panel shows the route network dataset generated by overline().

```r
plot(route_network, lwd=0)
plot(l, lwd = l$All / 10, add = TRUE)
lines(routes_fast, col = "red")
routes_fast$All <- l$All
rnet <- overline(routes_fast, "All", fun = sum)
rnet$flow <- rnet$All / mean(rnet$All) * 3
plot(rnet, lwd = rnet$flow / mean(rnet$flow))
```

To estimate the amount of capacity needed at each segment on the transport network, the `overline` function demonstrated above, is used to divide line geometries into unique segments and aggregate the overlapping values. The results, illustrated in the right-hand panel of Figure 1, can be used to estimate where there is most need to improve the transport network, for example informing the decision of where to build new bicycle paths.

Additional routing APIs were added with the functions `route_graphhopper`, `route_transportapi_public` and `viaroute`. These interface to Graphhopper, TransportAPI and the Open Source Routing Machine (OSRM) routing services, respectively. The great advantage of OSRM is that it allows you to run your own routing services on a local server, greatly increasing the rate of route generation.

## 2.5 Spatial interaction models

With its facilities for generating geographic travel data and native language of R, which excels in statistical modelling **stplanr** is well set-up to be a port of call for spatial interaction (SI) models. A brief example below shows the implementation of the *radiation model* in **stplanr**. Many other SI model forms could be added, perhaps using a generic function `si()`:

```
# load some points data
data(cents)
# plot the points to check they make sense
plot(cents)
```

```
class(cents)
```

```
## [1] "SpatialPointsDataFrame"
## attr(,"package")
## [1] "sp"
```

```
# Create test population to model flows
set.seed(2050)
cents$population <- runif(n = nrow(cents), min = 100, max = 1000)
# estimate
flowlines_radiation <- od_radiation(cents, pop_var = "population")
flowlines_radiation$flow
```

```
##  [1]          NA  28.5716171 199.1483930  45.6047446   7.5864754
##  [6]  21.5568487  41.2170441 105.2684494  24.7449354          NA
## [11] 118.5268678  10.1879935   4.3086722  57.4515754 259.3003676
## [16]  15.9131245  44.3764656 293.7173517          NA   9.8773421
## [21]   7.1271190  35.3137167  81.5452128  15.4279029  85.4088485
## [26]   5.9374332   8.8651941          NA  84.9504273  35.9825242
## [31]  26.8695174  53.6173692   1.8104829   1.2293437   0.7982433
## [36]  84.9504273          NA  19.6809596   2.7914752   1.2033517
## [41]  27.3885572 106.4446453  44.4324793  34.6086394 100.2158728
## [46]          NA 220.5555702  36.7618461  13.8249960 259.3003676
## [51]  59.2543662   6.1879080   3.8213690  34.8945124          NA
## [56]  22.4008006  10.4091206   4.4967751   1.7164989   1.3797507
## [61]   4.7760711   2.0211245 102.2283795          NA
```

```
sum(flowlines_radiation$flow, na.rm = TRUE) # the total flow in the system
```

```
## [1] 2937.987
```

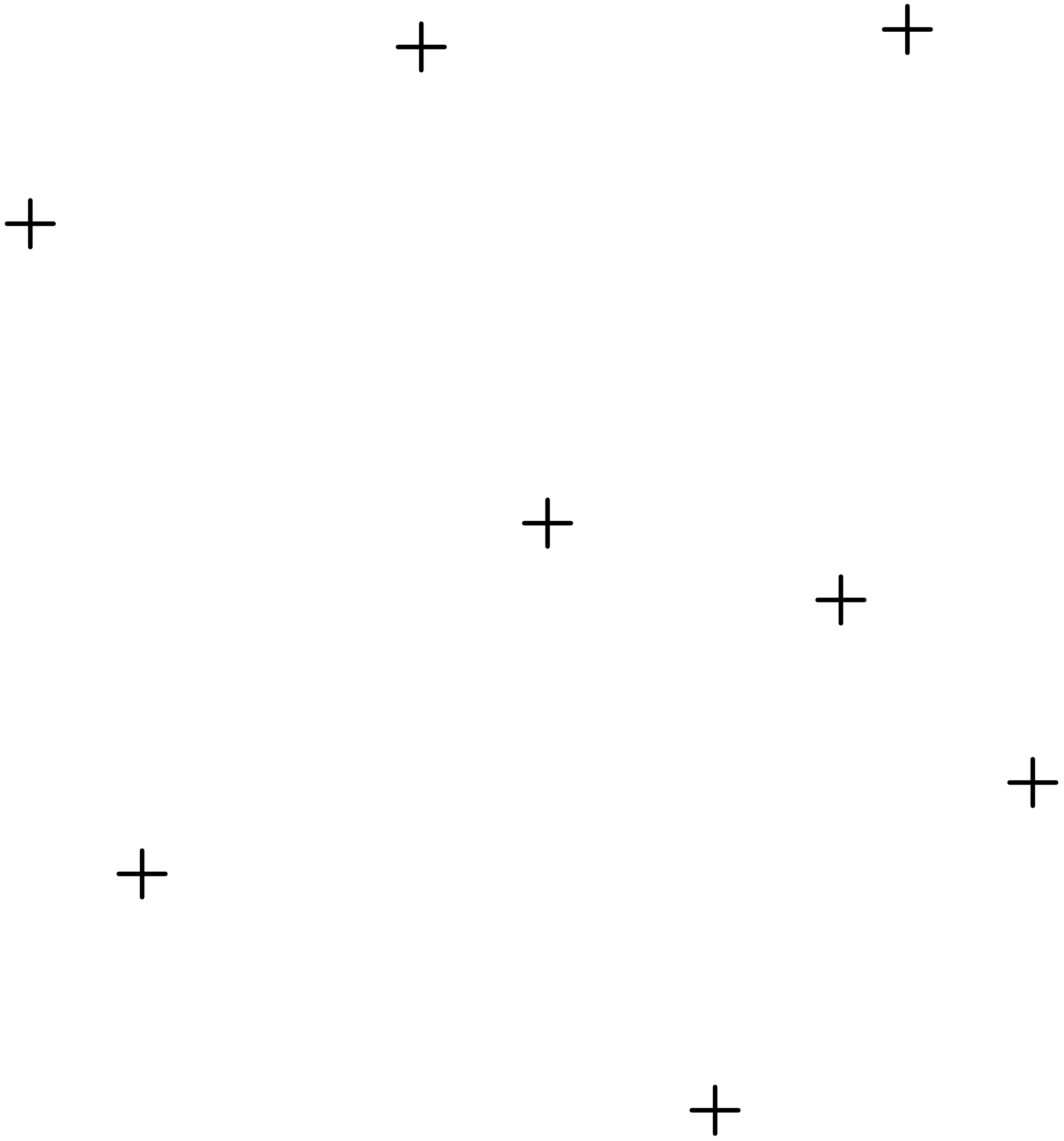

Figure 2: Results of running a Spatial Interaction model (the radiation model) with stplanr.

```r
sum(cents$population) # the total inter-zonal flow
```

```
## [1] 3450.191
```

```r
plot(flowlines_radiation, lwd = flowlines_radiation$flow / 100)
points(cents, cex = cents$population / 100)
```

The results show how packaging-up code into peer-reviewed, publicly accessible software, can enable the reproduction of results in the emerging field of transport data science. **stplanr**'s use in the DfT-funding Propensity to Cycle Tool (PCT) shows that software development can greatly assist with applied work (Lovelace et al. 2016).

The wider question is: what are the most desirable next steps (or cycle pedals) forward?

## 2.6 Next steps

The next step with **stplanr** is consolidation. Before moving to Version `1.0` we would like to update function names and behaviour based on first principles. Key to this will be user engagement and community collaboration so we will encourage this through 'hackathons', on-line demos, documentation, blogs and conferences such as GISRUK 2017, where this paper was presented.

## References

Lovelace, Robin, and Richard Ellison. 2017. "Stplanr: A Package for Transport Planning." *Under Review*. https://github.com/ropensci/stplanr.

Lovelace, Robin, Anna Goodman, Rachel Aldred, Nikolai Berkoff, Ali Abbas, and James Woodcock. 2016. "The Propensity to Cycle Tool: An Open Source Online System for Sustainable Transport Planning." *Journal of Transport and Land Use* 10 (1). doi:10.5198/jtlu.2016.862.

Ortzar, Juan de Dios, and Luis G. Willumsen. 2001. *Modelling Transport*. John Wiley; Sons.

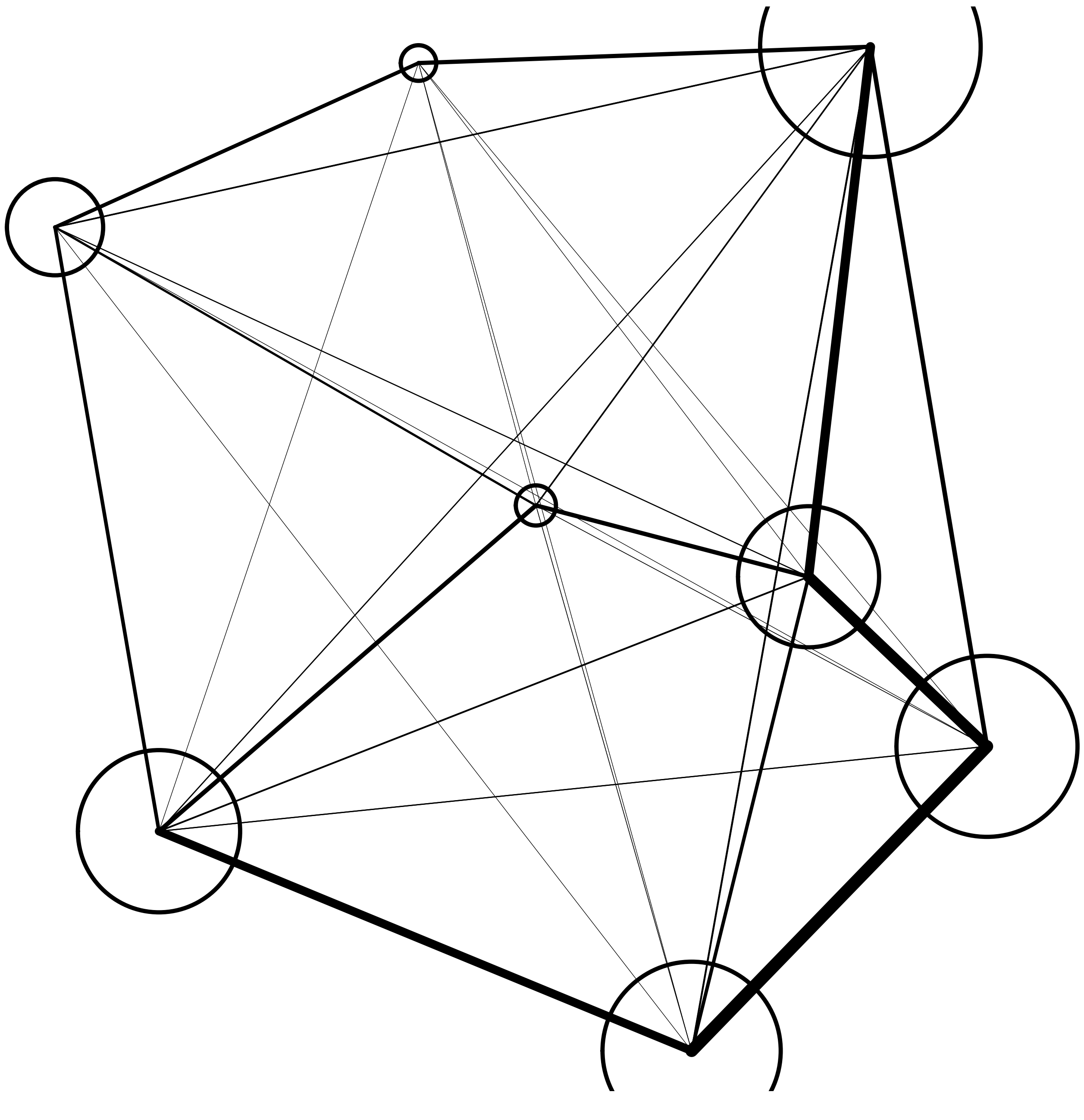

Figure 3: Results of running a Spatial Interaction model (the radiation model) with stplanr.